DOCUMENT RESUME

ED 397 267 CE 072 106

AUTHOR Hadipriono, Fabian C.; And Others
TITLE Safety in Construction Using Virtual Reality (SAVR): A Model for Labor Safety. Working Paper Series WP-022.
INSTITUTION Ohio State Univ., Columbus. Center for Labor Research.
SPONS AGENCY Ohio Board of Regents, Columbus.
PUB DATE Jun 96
CONTRACT OSURF-730237
NOTE 81p.
PUB TYPE Reports - Research/Technical (143)

EDRS PRICE MF01/PC04 Plus Postage.
DESCRIPTORS Building Trades; Computer Assisted Instruction; *Computer Graphics; *Computer Simulation; *Construction (Process); Models; *Occupational Safety and Health; *On the Job Training; Postsecondary Education; Risk Management; Safety Education; Safety Equipment; Teaching Methods; *Virtual Reality

ABSTRACT

An interactive training model called SAVR (Safety in Construction Using Virtual Reality) was developed to train construction students, novice engineers, and construction workers to prevent falls from scaffolding. The model was implemented in a graphics supercomputer, the ONYX Reality Engine2. The SAVR model provides trainees with an immersive, interactive virtual environment to perform "on-the-job" safety training without physically being at a real construction site. The model includes two major training environments: erection, which teaches trainees the correct procedure to erect a commonly used metal bracket form scaffolding; and inspection, which introduces several hazardous conditions in an existing platform and requires trainees to identify them visually. The development of SAVR involved four major tasks: knowledge acquisition, model development, model validation, and preparation of reports and manuals. Several common potential causes of falls from scaffolding platforms were chosen for the SAVR model, including component problems and connection problems. The second task, model development, included the construction of the three-dimensional graphical objects of the scaffolding components, the construction of the texture images for SAVR's interface panels, and the construction of the SAVR program. Construction used a developmental approach that included six steps: (1) defining the problem, (2) designing the solution, (3) refining the solution, (4) considering a testing strategy, (5) coding, testing, and debugging the program, and (6) documenting the program. SAVR demonstrates the potential of virtual reality technology in safety training using a safe environment. (Contains 45 references.) (KC)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED 397 267

Working Paper Series

CLR

Center for Labor Research

SAFETY IN CONSTRUCTION USING VIRTUAL REALITY (SAVR): A MODEL FOR LABOR SAFETY

Fabian C. Hadipriono,
Richard E. Larew, and
Ashraf S. Barsoum
Department of Civil and Environmental
Engineering and Geodetic Science
The Ohio State University

WP–022
June, 1996

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

S. Jordan

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

THE OHIO STATE UNIVERSITY

CE 072 106

BEST COPY AVAILABLE

ACKNOWLEDGMENT

This project was supported by grant OSURF No. 730237 from the Ohio Board of Regents through the Center for Labor Research at The Ohio State University. Dr. Charles J. Slanicka, the Director of the Center for Labor Research, and Dr. Warren R. VanTine, the Chair of the University Grants Committee, have been instrumental in providing this support. The contents of this report are solely the responsibility of the authors and do not necessarily represent the official views of the Center for Labor Research.

The authors wish to express their gratitude to the following professionals who evaluated and tested SAVR: Mr. David Kennedy (Assistant Area Director) Mr. Charles Sampsel (Compliance Safety and Health Officer), and Mr. Willie Robinson (Safety Specialist) from the United States Department of Labor—Occupational Safety and Health Administration; Mr. Norm Hughes (Director Sales Manager) of Economy Forms Company (EFCO), who supplied the drawings of the form scaffolding used in SAVR; Ms. Meg Conlon (Director of Safety) and Ms. Stacy McAllister (Safety Program Coordinate) from Builders Exchange of Central Ohio; Mr. Samuel C. Wright (Adult Learning Coordinator) from Performance Site Management in Ohio; and Mr. Bobby Reitter (Operations Manager) from Reitter Stucco Inc. Their comments and suggestions were essential to the improvement of the SAVR model.

Further appreciation is due to all the members of the Construction Laboratory for Automation and System Simulation (CLASS). Their comments and assistance throughout this project development were very valuable. James Tsay, a graduate student, assisted the authors in developing the VR environment. Finally, the authors wish to thank Mr. Bruce W. Rogers for his editorial work during the writing of this report.

TABLE OF CONTENTS

ACKNOWLEDGMENT i

TABLE OF CONTENT ii

LIST OF FIGURES v

EXECUTIVE SUMMARY vi

CHAPTER **PAGE**

I. INTRODUCTION

1.1 Background of the Study 1
1.2 Objectives of the Study 3
1.3 Tasks to Develop SAVR 4
1.3.1 Knowledge Acquisition 5
1.3.2 Models Development 5
1.3.3 Models Validation 5
1.3.4 Reports Preparation 5
1.4 Scope and Limitation 5
1.5 Organization of the Study 6

II. LITERATURE REVIEW

2.1 Introduction 7
2.2 Preliminary Studies of Construction Falls 7
2.3 Form Scaffolding Platforms 8
2.3.1 Causes of Falls from Form Scaffoldings 9
2.4 Virtual Reality 11
2.4.1 Types of Virtual Reality 12
2.4.2 Immersive Characteristics in Virtual Reality 13
2.4.3 Applications of Virtual Reality 14

III. SAVR CHARACTERISTICS

3.1 Introduction 16
3.2 The Training Environment 16

CHAPTER **PAGE**

3.3 Fall Causes in SAVR Scenarios 17
3.3.1 Component Problems 17
3.3.2 Connection Problems 17
3.3.3 Miscellaneous Problems 18
3.4 Interactive Training Scenarios in SAVR 18

IV. HARDWARE AND SOFTWARE FOR SAVR

4.1 Hardware 19
4.2 Software 19
4.2.1 World Tool Kit Development System 20
4.2.2 3D Studio 20
4.2.3 Microsoft Visual Basic 20
4.2.4 Utility Programs 20

V. IMPORTANT FEATURES IN WORLD TOOL KIT

5.1 Application Development Using WTK 22
5.1.1 Graphical Objects Construction 23
5.1.2 Texture Images Construction 25
5.2 The WTK Development System 27
5.2.1 The Universe Class 27
5.2.2 The Graphical Objects Class 28
5.2.3 The Sensors Class 28
5.2.4 The Viewpoints Class 28
5.3 The Major Tasks in An Application Using WTK Functions 29
5.3.1 Initializing the Simulated Environment 29
5.3.2 Specifying the Interactive Scenarios During the Simulation 30
5.3.3 Starting the Simulation Loop and Performing the Simulation 30
5.4 Types of Simulated Graphical Objects 31
5.5 The Object Task Function 32

VI. THE CONSTRUCTION OF SAVR'S PROGRAM

6.1 Development Approach 34
6.1.1 Defining the Problem 34
6.1.2 Designing the Solution 34
6.1.3 Refining the Solution 35
6.1.4 Considering A Testing Strategy 36
6.1.5 Coding, Testing, and Debugging the Program 36
6.1.6 Documenting the Program 36
6.2 Programming Considerations 37
6.2.1 The Programming Environment 37
6.2.2 The Structure of the Program 37

CHAPTER **PAGE**

6.2.3 The Flow of the Program 38
6.2.4 The Graphical-User Interface 39
6.2.5 The Potential for Expanding the Application 42
6.3 The Design and Implementation of SAVR's Interactive Scenarios 42
6.3.1 The Design and Implementation of the Erection Interactive Scenarios 42
6.4 Techniques for Developing a User-Friendly Graphical-User Interface 43
6.4.1 Design and implementation of SAVR's Control Panels 43
6.4.1.1 Fixing the Projection of SAVR's Control Panels 44
6.4.1.2 Activating SAVR's Interface Buttons 46
6.4.2 Design and Implementation of SAVR's Scoring System 48
6.5 Model Transformation from Onyx to PC 49

VII. EVALUATION AND TESTING

7.1 Introduction 51
7.2 SAVR Evaluation and Testing 51
7.3 Formal Evaluation 52
7.4 Results of Formal Evaluation 52

VIII. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

8.1 Summary 59
8.2 Conclusions 60
8.3 Recommendations 62

APPENDICES

A REFERENCES 63
B ABBREVIATIONS 68

LIST OF FIGURES

FIGURE	PAGE
2-1 Metal bracket form scaffolding modeled after EFCO	9
2-2 Components used for a fault tree model of falls from form scaffolding	10
2-3 Six degrees of freedom for position and orientation tracking	13
3-1 Scaffolding components as shown in a virtual environment	16
5-1 WTK application development process	22
5-2 A 3D Studio triangular representation of faces	24
5-3 A quad representation of faces	24
5-4 A graphical model of a steel bracket component modeled after EFCO	25
5-5 The image applied to the surface of the inspection interface panel	26
5-6 The default simulation loop	31
6-1 SAVR's main modules	35
6-2 The flow of the SAVR program	39
6-3 A snap-shot from SAVR's start module	40
6-4 A snap-shot from SAVR's erection module	41
6-5 A snap-shot from SAVR's inspection module	41
6-6 Fixing the projection of SAVR's control panels	45
6-7 The values used to calculate the mouse position relative to the WTK Window	48
6-8 An image captured from the PC environment	50
6-9 An image captured from the PC environment	50
7-1 System evaluation form	54
7-2 Detail of graphical models	56
7-3 Appearance of graphical models	56
7-4 The system's user interface	57
7-5 The system's problem/solution representation	57
7-6 The system's applicability for training	58
7-7 The system's overall performance	58

EXECUTIVE SUMMARY

In this study, an interactive training model called SAVR was developed to train construction students, novice engineers, and construction workers to prevent falls from form scaffolding. The model was implemented in a graphics super computer, the ONYX Reality Engine2. The SAVR model provides trainees with an immersive, interactive virtual environment to perform "on the job" safety training without physically being in a real construction site. This model includes two major training environments: erection, which teaches trainees the correct procedure to erect a commonly used metal bracket form scaffolding; and inspection, which introduces several hazardous conditions in an existing platform and requires trainees to visually identify them.

The development of SAVR involved four major tasks: knowledge acquisition, model development, model validation, and preparation of reports and manuals. Several common potential causes of falls from form scaffolding platforms were chosen for the SAVR model, including component problems and connection problems. The second task, model development, included the construction of the 3D graphical objects of the scaffolding components, the construction of the texture images for SAVR's interface panels, and the construction of the SAVR program. In constructing the SAVR program we adapted a developmental approach that included six steps: (1) defining the problem, (2) designing the solution, (3) refining the solution, (4) considering a testing strategy, (5) coding, testing and debugging the program, and (6) documenting the program.

SAVR demonstrates the potential of virtual reality technology in safety training using a safe environment. We expect that the interactive visualization in the SAVR model will enable trainees to visually memorize the erection and inspection patterns and consequently, apply them in real-life situations.

The interactive visualization in SAVR called for an extensive use of graphics due to the complexity of the 3D objects presenting the steel components of the scaffolding. While these objects were constructed using a minimal number of polygons, they still involved complex graphics which require advanced rendering capabilities. Accordingly, SAVR was developed on a graphics super computer to facilitate a real time interaction.

The modular design for implementing the start, help, erect and inspect environments in the SAVR model enables future expansion of the model to include additional types of scaffolding platforms, helped in avoiding a significant decrease of the frame rates, and simplified the debugging process during application development. Furthermore, the implementation of an interface panel in each module and the use of textures for panel functions provided a user-friendly graphical-user interface.

Throughout the project period (12 months), SAVR was continuously tested and evaluated by its developers (investigators and his student researcher), particularly for its erection and inspection modules. Enthusiastic comments by experts indicate that SAVR can become a self-contained training tool for entry level construction laborers. Thus, the SAVR system is expected to contribute to the avoidance of construction falls, and subsequently, to the reduction of injuries and fatalities of construction laborers during construction operations.

Based on our experience in SAVR development, we recommend incorporating more scaffolding types for worker training and enhancing the rendering speed of the super computer.

CHAPTER I
INTRODUCTION

1.1 Background of the Study

The construction industry is one of the largest industries in the United States, making up about 10% of the country's GNP. Construction laborers have established themselves as one of the country's largest work forces. However, the worker injury rate in the construction industry is 54% higher than the rate for all industries, making construction one of the most hazardous occupations [TBR 1990]. Construction worker accidents have been reported in numerous articles [NIOSH 1988, NIOSH 1989a, OBWC 1989a]. Moreover, the statistical compilations of such accidents have been presented in many reports [BLS 1986a, BLS 1986b, NSC 1987, OBWC 1989b]. Regulations and codes to avoid these accidents have also been published [AGC 1990, CFR-29 1988, OBWC 1989a].

Despite the above and the Occupational Safety and Health Administration's (OSHA's) stricter rules and higher penalties for repeat offenders, there is no indication of a reduction in labor-related construction accidents [Hadipriono and Diaz 1988]. In 1990, OSHA identified the most common types of fatalities involving construction workers: falls (33%), struck-by object (22%), caught-in-between objects (18%), electric-shock (17%), and other (10%) [ENR 1991]. The study, which was based on 3,496 construction fatalities recorded from 1985-1989, clearly shows that falls are the paramount cause of these fatalities. Numerous studies [BLS 1986a, 1986b, and 1988; NIOSH 1989a and 1989b; NSC 1987; TBR 1989a, 1989b, 1990] have also suggested that falls constitute the largest percentage of construction fatalities and injuries. The US Department of Labor reveals that falls are "one of the leading causes of traumatic occupational death, accounting for 8% of all occupational fatalities from trauma in 1986" [BLS 1988]. The National Institute for Safety and Health (NIOSH) National Traumatic Occupational Fatality (NTOF) data base shows that during 1980-1985, falls represented about 10% of all traumatic occupational deaths [NIOSH 1989a and 1989b]. In its *Accident Facts*, the National Safety Council [1987] reveals that by type of accident, falls represent 12% of all deaths in work accidents, the second highest type after motor-vehicle accidents. In the same report, a study about work injuries by type of accident in 1983 shows that construction falls were one of the highest types, representing 21.4% of all construction accidents.

From an international perspective, the Health and Safety Executive (HSE) of the United Kingdom reported that about half of the fatalities in construction operations were

attributed to falls from elevations [IC 1994]. In Korea, although the statistics of falls are not available, in a recent study of construction accidents, the Korean Industrial Safety Institution reported that these accidents are on the increase [Ann 1993]. Further, the institution criticized the construction industry for not adequately stressing the importance of protecting their workers from accidents. In Japan, the Nikkan Kensetsu Kougyou Shinbun [1992] reported that in 1990 alone about 60,000 severe construction accidents resulted in 1,075 labor deaths, representing nearly 40% of the fatalities of all industries combined. It is no surprise that the Japanese nicknamed their construction industry as the "3K industry," referring to *kiken* (dangerous), *kitanai* (dirty), and *kitsui* (difficult).

Among these falls, scaffolding-related accidents have been found to be the most frequent type of falls. In 1986, a study was conducted to investigate construction accidents of 85 major construction accidents involving concrete structures that occurred around the world over the past 23 years [Hadipriono and Wang 1986]. Many of the results reveal the significant role of scaffolding-related accidents. Bobick et al. [1990a, 1990b] from NIOSH's Division of Safety Research discussed numerous cases of fatal falls from scaffoldings. To illustrate the negative impact of a scaffolding accident, in 1978, the top part of the West Virginia Cooling Tower fell from a height of 166 feet, bringing down the scaffolding and workers and causing the deaths of 51 laborers. This accident is considered the worst construction disaster in U.S. history. Millions of dollars were lost, contractors went bankrupt, and the project was completed months behind schedule [Hadipriono 1985].

Several of the above studies have revealed that inadequate training--which in turn lead to on-site ignorance and negligence--is an underlying cause of these accidents. Walker [1981] and Shaw [1981] from the Institution of Structural Engineers in the United Kingdom were among the first to conduct an extensive survey of the underlying causes of hundreds of construction and structural accidents. The survey reveals a close relation between these accidents and the inadequate training of young engineers and construction laborers, which, in turn, led to ignorance and negligence. Young engineers, who have never worked at a job site are often insufficiently aware of the constant danger construction laborers face during a construction operation, and they often ignore and neglect the safety of their workers.

Later studies [Hadipriono 1992b, Hadipriono and Larew 1991] have confirmed such findings. Furthermore, they have indicated that, unlike other industries, construction is unique in that its operations are site-oriented, complex, multi-faceted, and often unprecedented. Therefore, most construction engineers and workers need to spend many years in the field in order to assimilate an adequate knowledge about actual construction operations. Yet, because of the development of new materials, equipment, and construction approaches, they must have a more scientific background and achieve sufficient training skill at an accelerated pace. Engineers and workers alike need to know more than ever and in less time than ever--they cannot rely on years of field experience anymore. Traditional construction training alone is insufficient to warrant the reduction of on-site ignorance and negligence.

Both private and public sectors have recognized the urgent need to improve labor training. Innovative substitutions for traditional field experience have been introduced to improve the nation's labor training system. Despite these efforts, on-site ignorance and negligence remain, leading to construction accidents and, subsequently, causing time delays, cost overruns and, very often, bankruptcy of companies. Indeed, the construction industry has the highest bankruptcy rate of all industries [Hadipriono and Larew 1985].

1.2 Objectives of the Study

To effectively overcome the above problem, the proposers have developed SAfety in construction using Virtual Reality (SAVR). SAVR is expected to fulfill two complementary goals: (1) to pioneer research using a new technology that will be available in the near future, and (2) to use revolutionary computer development to accelerate safety training for construction laborers.

In this project, SAVR models were developed to incorporate the most common construction accidents, *construction falls*, i.e., unintentional falls during construction from higher elevations. The working platform considered here is limited to *form scaffolding*, a temporary structure attached to the frame of a formwork through the use of brackets. In structuring SAVR's algorithm, we maintained a flexible structure to allow future expansions to include other scaffolding platforms, which will be treated separately and become an extension of SAVR. The form scaffolding models were developed in a workstation (UNIX-based).

Specifically, SAVR provides construction trainees with the following:

(1) A tool for "on the job" training on simulated construction platforms, one of the most hazardous workplaces in all industries. SAVR trains construction laborers through the repetitive fact-finding process, virtual reality (VR) models are used for teaching domain-dependent facts derived from the heuristic and experiential judgments of safety experts. Here, trainees act as construction workers inspecting hazardous platforms, detecting each hazardous condition, and eliminating the condition, without actually being there. The repetitive "on-the-job" learning experience is expected to narrow the experiential gap between a novice and an expert. Hence, SAVR models created new intuitive understanding about the dangers in construction operations.

(2) A tool to develop the ability to think creatively, solve problems accurately, and learn analytically at an accelerated pace. SAVR's immersive nature allows trainees to explore and establish relationships among symbolic elements in a first-person environment. Consequently, SAVR accelerates trainees' skill to perform safe construction operations and to obtain the practicality and feel for engineered construction, engineering judgment, and understanding of underlying causes of accidents. This, in turn, leads to the reduction of potential ignorance and negligence on construction sites.

In addition, SAVR is expected to expose users to concepts and/or techniques that were not previously possible. More specifically, the anticipated significance of SAVR can be explained as follows:

(1) SAVR multisensory learning attributes allow users to enter a symbolic space and immerse themselves in symbolic elements, thereby providing them with the "feeling" of being present at a construction site. Trainees virtually "participate" in on-site safety inspection through exploratory and navigational attributes. Exploration reveals expected and unexpected alternatives and allows trainees to identify construction and safety-related information and information-complexes. Navigation allows trainees to return to regions known to have desired information. With these attributes, the SAVR learning process is anticipated to remove cognitive barriers to variables that are continuous and dynamic in nature, and that are especially valuable in providing or augmenting cognitive access to complex information (e.g., data/knowledge regarding unprecedented and/or multi-faceted construction operations).

(2) SAVR provides an example of both short and long term solutions to many construction problems. In the short term, SAVR is a safe and cost-effective means to create a self-contained interactive training environment, and is expected to revolutionize the concept of learning, cognition, and problem-solving with respect to safe and practicable construction operations. In the long run, SAVR implementation and dissemination will minimize on-site ignorance and negligence and contribute to a safer construction environment. In turn, this will result in the saving of costs, time, and lives.

It is expected that users (trainees) can use SAVR as means to quickly expand learning capacity, enhance reasoning abilities, and compensate for user limitations, so as to minimize ignorance and negligence on construction sites.

To meet the above expectations, SAVR furnishes trainees with the experience provided by a cognitive model of the dynamic change process, such as that offered by VR technology [Hadipriono 1992a]. Fisher [1991] describes the VR technique as an "innovative way to represent first-person or direct experience through the development of multi-sensory media environments in which viewers can interact with the information presented as they would in encountering the original scene." Despite the recent proliferation of VR applications in numerous fields, neither R&D nor applications of VR techniques are currently taking place in construction, not even in an effort, first, to train laborers and young engineers to perform construction work safely and then, eventually, to save cost, time, and human lives. Hence, the development of SAVR as a training tool is particularly timely.

1.3 Tasks to Develop SAVR

In order to develop SAVR, four tasks were carried out: (1) knowledge acquisition, (2) model development, (3), model validation, and (4) reports preparation.

1.3.1 Task-1: Knowledge Acquisition

This task included gathering information from experts, literature, OSHA codes, OBWC codes, major scaffolding companies, and related research. In recent research, Hadipriono et al. [1995b] compiled and represented the knowledge required to develop an expert system (Safety First) to investigate construction accidents. The results of this research are discussed Chapter II of this report. This knowledge includes significant causes of construction falls from form scaffolding.

1.3.2 Task-2: Model Development

Among all of the tasks involved in SAVR development, this task consumed the greatest amount of time and effort. Two models were developed for (1) inspection and (2) erection of the form scaffolding. The inspection model allows users to inspect the problem associated with the laborer's safety when working on the scaffolding, while the erection model permits users to train themselves by using the VR instrument to erect the form scaffolding. Model development included the construction of the 3D graphical objects and sound files which are used in the simulation, and the development of an algorithm for SAVR to immerse trainees in an interactive simulation using the VR instrument. Implementation of this task is described in Chapters III, IV, V, and VI.

1.3.3 Task-3: Model Validation

Testing and evaluation of SAVR by the system developers (the investigators and their research assistants) were performed throughout the development of the models. When the inspection model was completed, it was validated through a formal testing process by eight construction safety professionals. The validation process and results are presented in Chapter VII.

1.3.4 Task-4: Reports Preparation

The reports for this study consist of one progress report and this final report.

1.4 Scope and Limitations

SAVR was developed to train novice engineers and construction workers to detect hazardous conditions that may lead to construction falls from form scaffolding platforms. It immerses trainees into an interactive simulated environment of a construction site in which there is a concrete wall under construction bounded by a frame of formwork. A commonly used form scaffolding platform is attached to this frame of formwork. Trainees can visually inspect the form scaffolding platform to find the potential causes of falls. SAVR also provides a visual simulation to show the proper erection procedure. In addition, sound files were developed to teach new users how to use the program and assist them during the simulation. These sound files enhance SAVR's immersive capability.

The potential causes of falls in SAVR were limited to those which can be visually detected, such as a missing guardrail component from the guardrail system. Causes which cannot be visually detected, such as erroneous design of scaffolding sections, were not

included. Furthermore, only high frequency visually detectable causes were included. Other causes may become an extension of SAVR in the future.

1.5 Organization of the Study

This study consists of seven chapters and two appendices. Chapter I includes the background of the study, objectives, tasks, benefits, and scope and limitations. Chapter II, the literature review, discusses construction falls from form scaffolding. It also provides background information related to virtual reality and some of its applications. In Chapter III, fall characteristics in SAVR are explained in detail. The hardware and software which were used to develop SAVR are described in Chapter IV. Chapter V introduces the important features of the major software in SAVR development, World Tool Kit (WTK) development system. Chapter VI explains the development of SAVR's program. This includes the developmental approach, programming considerations, and techniques for developing SAVR's graphical-user interface. The testing and evaluation of SAVR are elaborated in Chapter VII. Finally, Chapter VIII includes the summary, conclusions, and recommendations.

CHAPTER II
LITERATURE REVIEW

2.1 Introduction

In addition to the knowledge gathered from experts and literature, information from previous research conducted at the Construction Laboratory for Automation and System Simulation (CLASS) was used for developing SAVR. This previous work (Section 2.2) involved several studies on construction falls which provided a solid background for this study. Furthermore, our past experience in developing VR applications (Sub-Section 2.4.3) accelerated the research process and consequently, allowed more time for refinement and fine tuning of the final product.

2.2 Preliminary Studies of Construction Falls

In 1986, a study was conducted to investigate 85 major construction accidents of concrete structures that occurred around the world over the past 23 years [Hadipriono and Wang 1986]. Many of the results reveal the significant role of construction falls in scaffolding failures. In another study sponsored by the Center for Labor Research at The Ohio State University (CLR-OSU), Hadipriono [1992b] performed a survey of the internal and external causes of falls. Also, the feasibility of integrating a fault tree system and an expert system to improve labor safety was established. A modified fault tree system was used in this study. The knowledge used to determine the causes of falls and information leading to these causes was obtained from research literature and interviewing experts. The study was limited to unintentional construction falls from elevated floor openings.

The results of the study confirm an earlier prognosis of using fault tree diagrams to best represent the structure of expert knowledge. These fault tree models describe casual relationships and determine the minimum number of combinations of causes (minimum cut set) of construction falls. Hence, the fault trees can be used as a "road map" of the expert knowledge when diagnosing the cause(s) of a fall. In addition, these fault trees were used as a guide in developing the user interface (consultation modes) in a simple and logical way. The expert system has the capability to simulate the heuristic reasoning process of an expert. The knowledge acquired based on experts' heuristic judgment was incorporated into the system. Through its interface mechanism, the expert system matched evidence with the knowledge compiled in the knowledge base in order to obtain the solution. As a result of the study, a report was published and presented to the

Ohio Board of Regents [Hadipriono 1992b] and two papers were published in the American Society of Civil Engineering (ASCE) journal [Hadipriono 1992c, Hadipriono 1992d].

The CLR-OSU sponsored study has led to a three-year grant from the National Institute of Safety and Health (NIOSH). In this study, a more refined expert system of construction falls was developed [Hadipriono et al. 1995c]. The construction falls were classified as falls from higher elevation, falls from same elevation, and slips (without falling). The platforms from which falls could occur were categorized into floor openings, roofs, steel beams, top of walls, and ladders. A separate study was conducted for falls from scaffolding. We considered the worker and the platform as components of a system. We analyzed all possible events that could cause the fall. The general causes of falls were classified into enabling (internal problems), triggering (external active), and loss of supporting components (external passive). In this study, the fault tree expert system was extended to include graphics components. At this stage, this ongoing project has resulted in two master theses [Vargas 1993, Yoo 1994] and a paper published in the Associated Schools of Construction (ASC) conference proceedings [Vargas and Hadipriono 1995].

2.3 Form Scaffolding Platforms

A form scaffolding is a temporary structure attached to a frame of formwork through the use of brackets. This structure provides a working platform to inspect the concrete form and re-bars, and to support workers during casting the concrete. The OBWC [1979] identifies three major types of form scaffoldings: wooden and metal brackets. In this study we only consider the metal bracket, which is commonly used in construction sites.

The metal bracket form scaffolding (Figure 2-1) consists of steel brackets, planks, and a guardrail system. The brackets are used to attach the platform to the main formwork and to support the planks. The planks are the working surface that will support the workers. The guardrail system is used to protect the worker from falling from the platform. The guardrail system consists of vertical posts that are attached to the brackets. These posts support three horizontal members: a handrail, an intermediate rail, and a toe board. All components of the form scaffolding should be designed and erected to support a minimum of four times the maximum rated load without failure [OBWC 1979].

The bracket section typically consists of three steel members: a ledger, diagonal, and vertical. These members are usually welded or bolted to assemble the bracket section. The ledger transfers the loads from the planks to the diagonal and the vertical members. The vertical member transfers the loads from the diagonal and ledger to the main formwork through a stringer or a waler. The maximum allowable spacing between the brackets is 8 feet (2.4 m) measured from the center of one bracket to the center of another. Bracket attachments to the main formwork are usually done using bolted connections; however, welded connections are also allowed [OBWC 1979].

As for the connection between the bracket and the main formwork, the post can be bolted or welded to the bracket section. The maximum spacing between the posts should

not exceed 8 feet (2.4 m). The horizontal guardrail members should be bolted to the post. The planks can be bolted to the ledgers or overlap the ledgers by a minimum of 6 inches (152 mm) each way at the intermediate ledgers and a maximum of 12 inches (305 mm) at the end ledger [OBWC 1979]. The planks must be scaffold grade wood planks of a minimum cross section 2x9 inches (51x229 mm). The minimum cross sections of the handrail, intermediate rail, and toe board are 2x4 inches (51x102 mm), 1x6 inches (25x152 mm), and 1x4 inches (25x102 mm) respectively. The handrail should be installed at 42 inches (1100 mm) above the working surface with a tolerance of plus or minus 3 inches (76 mm). The intermediate rail should be installed in the center of the vertical distance between the handrail and the toe board. The causes of falls from form scaffolding platforms are discussed below.

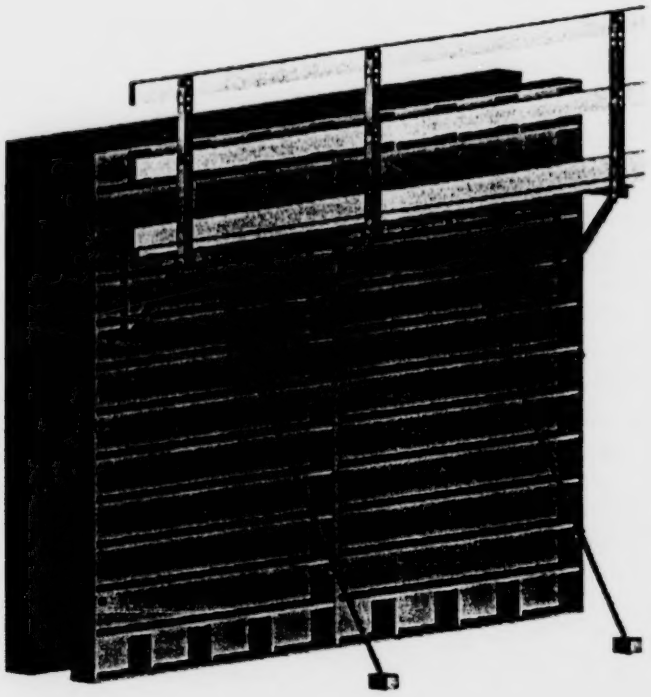

Figure 2-1. Metal bracket form scaffolding modeled after EFCO

2.3.1 Causes of Falls from Form Scaffoldings

Form scaffoldings are elevated platforms and falls from them are classified by OBWC [1979] as falls from higher elevations. However, workers may slip or fall on the

BEST COPY AVAILABLE

platform, which is classified as slips and falls from the same elevation. In this study, we included only the falls from higher elevations. The causes of such falls can be numerous and are dependent on many variables and conditions [Hadipriono et al. 1995b].

In the earlier study by Yoo [1994], fault tree models were developed to identify the possible causes of falls from form scaffolding. This study focused on causes that are attributed to the components of scaffolding sections; however, the causes that are related to the worker were not analyzed. Recently, Hadipriono et al. [1995b] included the potential causes that are attributed to the workers. In both studies, developing the fault tree models was based on the assumption that the worker and the components of the scaffolding platform form an integrated system (Figure 2-2) and then, the possible causes of its failure were analyzed. The first component of the system is the worker who is supported by the planks. The planks are supported by the brackets, which in turn, are supported by the main formwork.

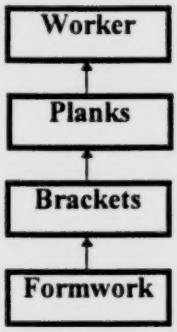

Figure 2-2. Components used for a fault tree model of falls from form scaffolding [Hadipriono et al. 1995b]

In the above system it is assumed that the worker falls due to any failure of any supporting component, such as the planks, brackets, or main formwork. The failure of any component of the system was attributed to three type of causes: enabling, triggering, support-related. In addition, the worker may fall due to problems with the safety devices, which were considered as safety conditioning causes. The following sub-sections generally discuss the enabling, triggering, support-related, and safety conditioning causes based on the work done by Hadipriono et al. [1995b]. (Section 3.3 provides detailed information on the fall causes which are considered in SAVR.)

Enabling causes include the internal conditions which may lead to a fall accident. This type of causes can be related to each component of the system. For example, the bracket enabling causes include inadequate design, inadequate erection, and defective

bracket section. Other enabling causes can be related to the worker, such as a worker's health problem or negative attitude towards safety rules. An enabling cause may lead to a fall either by itself or combined with another cause(s), such as a triggering cause.

Triggering causes are external conditions which also may trigger a fall accident, such as an external impact caused by equipment. These types of causes can also be related to each component of the system, such as the worker, planks, and brackets. For example, bad weather conditions such as strong winds may affect the worker's balance and cause the worker to fall. Such an accident is usually contingent upon the failure of the guardrail system to prevent the fall. Another example is the failure of the brackets due to equipment impact, which would lead to failure of the planks supporting the worker, and subsequently, causing the worker to fall. The failure of planks in this case is considered a support-related cause.

Support-related causes are the most common causes of falls from scaffolding platforms [Hadipriono et al. 1995b]. These causes are only related to the supporting components in the system (planks, brackets, and formwork). The failure of any supporting component is expected to ultimately cause the worker to fall. This failure may be due to the enabling and/or triggering causes discussed above.

Safety conditioning causes are related to the safety devises which are required to prevent the worker from falling. In the case of the form scaffolding platform, the only requirement is the guardrail system. Failure of the guardrail components is attributed to the related enabling and/or triggering causes.

2.4 Virtual Reality

Virtual reality (VR) has been defined from different perspectives and using different terminology. Pimental and Teixeira [1993] defined VR as "...a tool for revealing new ways of looking at information. VR gives users an efficient and effortless flow of data, details, and information in the most natural format possible--vision, sound and sensations presented as an environment, part of the natural media of human experience and thought." Another definition of VR is "...a way for humans to visualize, manipulate, and interact with computers and extremely complex data." [Aukstakalnis and Blatner 1992]. Simply put, VR is a sophisticated computer-human interface that employs advanced technology such as computer graphics and data communications to generate a simple and efficient way for interaction.

The use of VR technologies for simulation is explained by Heilig [1962], the inventor of what is considered a pioneering VR model, the Sensorama Simulator:

> "The present invention, generally, relates to simulator apparatus and, more particularly, to apparatus to stimulate the senses of an individual to simulate an actual experience realistically. . . . There are increasing demands today for ways and means to teach and train individuals without actually subjecting the individuals to the hazards of particular simulations. . . . Accordingly, it is an object of the present invention to provide an

apparatus to simulate a desired experience by developing sensations in a plurality of the senses."

In his recent book *Virtual Reality*, Rheingold [1991] asserts that humans do best in cognitive simulation through VR model-making. He further states that computation and display technology converge on hyperreal simulation capability; computer simulations will become so realistic that people will not be able to differentiate them from non-simulated reality.

2.4.1 Types of Virtual Reality

In VR applications, humans interact with a computer-generated environment (known as a virtual environment or virtual world) which exists in the computer memory. The environment is presented (projected) to the user through one or more output devices that the user interacts with through input device(s). However, not all VR environments are interactive; the level of interaction in VR environments can be grouped into three basic types: passive, exploratory, and interactive [Aukstakalnis and Blatner 1992].

In the passive type, a user cannot change the virtual environment. An example of passive VR is watching a 3D animation, in which 3D graphics models are animated using a specific scenario to show a particular behavior. In such an animation, the projection of the virtual environment depends on a predetermined position and orientation of the user (viewer). A viewer can neither change any behavior of the graphics models nor the projection of the environment.

The second type of VR interaction is the exploratory one, in which a user can explore the virtual environment, but cannot change the behavior of its components (building blocks). Exploring the virtual environment is done by allowing the users to walk through the virtual environment, which changes their position and orientation, and consequently, changes the projection of the environment. For example, a user can walk through a virtual location representing a construction site and explore the building blocks of the scene, such as the blocks of the structural elements under construction, formwork structures, and scaffolding platforms. However, these building blocks cannot be manipulated.

The most advanced and powerful type is the interactive environment. In this type, the user can explore the building blocks of the environment and manipulate them as well. For example, in the previous case of the construction site, the user can explore the scaffolding sections in the site, detect a defective element, and replace it. That would involve allowing the user to walk-through the scene, choose a defective element, remove it from the environment, and add the proper element to the environment. Generally, such a manipulation process depends on the features of the application that are implemented by the programmer(s) during application development. Interactive VR may also be immersive, which adds more power to its user-machine interface.

2.4.2 Immersive Characteristics in Virtual Reality

Immersive characteristics in VR were explained by Aukstakalnis and Blatner [1992]. In their daily life, humans are immersed in real 3D space, but in VR the space is computer-generated 3D space, i.e. virtual space. VR uses special technology to simulate the state of life-like immersion as closely as possible. In VR applications, immersion includes three main elements: depth perception, position and orientation, and interaction.

The feeling of space depth is accomplished by using natural senses as we do in our daily life. In VR, this sensation is gained through seeing, hearing, and feeling using special peripherals. For example, in using special goggles such as a head-mounted display (HMD), we only see the surrounding virtual 3D space generated by the computer; we are not able to see the real surrounding space. We can partially feel the 3D space and identify the objects in that space, i.e. their color, and the position in which they are located. By adding another peripheral such as a 3D sound system we can identify the sounds and their locations and thus be more immersed in the virtual space. In order to explore the space, we change our position and orientation.

The second immersion element is related to the position and orientation of a person (viewer) immersed in the virtual environment. As in real life, people expect to see an object closer if they move towards it, they expect the same while immersed in the virtual environment. The computer tracks the position and orientation of the viewer and updates the 3D space accordingly. Position and orientation tracking use a total of six degrees of freedom; three degrees represent the movement relative to the x, y, and z coordinates and the other three represent the rotational directions roll, pitch, and yaw. (Figure 2-3 shows the six degrees of freedom that are used for position/orientation and tracking.)

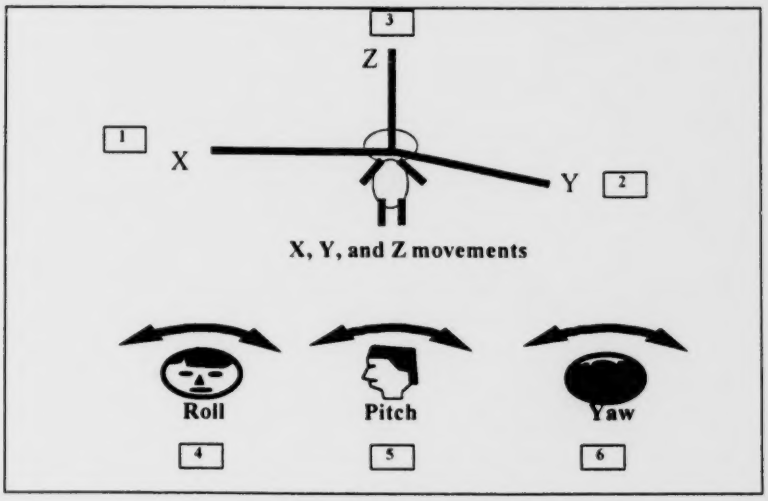

Figure 2-3. Six degrees of freedom for position and orientation tracking [Aukstakalnis and Blatner 1992]

The third immersion element is interaction. In the previous two elements we could feel the depth of the space and walk through it. In the third element we can interact with objects in the virtual environment as we do in the real world. For example, we should be able to pick up and move objects and feel their texture.

Immersion in VR environments allows users to interact with computers in a simple way; they can explore many spaces and manipulate many objects without having to memorize and write a special code for each task. However, programmers have to do more work to develop VR applications. They have to keep up with the rapidly advancing technology used in the VR industry. The industry is still growing fast and many significant applications have just been developed in the last few years covering different research and application areas.

2.4.3 Applications of Virtual Reality

VR techniques have been used in numerous domains, such as landscape and architecture [Goldman and Zdepski 1991], and education [McCluskey 1991, Dowding 1991], and entertainment [Glenn 1991, Pimentel and Teixeira 1993, Jacobson 1994]. Engineering applications of VR models include a project by NASA to visualize computational fluid dynamic data in a virtual wind tunnel [Aukstakalnis 1993], research by the U. S. Army Corps of Engineers, at Vicksburg, Mississippi, to simulate water flow through a rip rap test channel [Aukstakalnis 1993], and a study by the Fokker aerospace company in Europe to simulate the remote manipulator arm of the Hermes space shuttle [Coull 1991].

In health and medicine, VR is used for training medical students about human anatomy and for surgical procedure practice. In education, flight simulators have been frequently used for training pilots. New pilots are trained to overcome critical weather and other dangerous conditions. Also, military pilots can experience using new airplane models and fighting techniques using sophisticated simulators in a safe environment and with a training cost that is much lower than traditional methods [Aukstakalnis and Blatner 1992].

For the past three years, we have been using VR techniques in our research projects. The first project, involving accident response, was sponsored by The Ohio State University Center for Intelligent Transportation Research (OSU CITR). A traffic accident often results in a simultaneous response from various units, such as highway patrol, paramedic, and/or firefighter units, which in turn, leads to traffic congestion, or an erroneous response may lead to a delay in rescuing the victim(s). The INtelligent TRaffic Evaluator for Prompt Incident Diagnosis in a Virtual Reality environment (INTREPID-VR) was developed furnishes users with a way to experience "being present" at the accident scene, and thereby avoiding inadequate accident verification and erroneous assessment. The results of the study were presented in the Fourth International Conference on the Application of High Performance Computers in Engineering [Hadipriono et al. 1995a] and published in the proceedings of the second International Conference on Visualization and Intelligent Design in Engineering and Architecture

[Hadipriono et al. 1995b]. A demonstration of the system at the Intelligent Transportation System Ohio Annual Meeting Survey [Barsoum et al. 1995] won a second place.

An unfunded research project concerning construction falls was conducted as a master thesis [Soedarmono 1994]. The knowledge about construction falls from our earlier studies was used to develop a 3D model of a four-story building under construction. The model displays a concrete frame of a building with several platforms, such as floor openings, floor edges, wall openings, unfinished walls, roofs, and ladders. These models were successfully constructed. They have been evaluated by a safety expert from a major construction company on a limited basis with satisfactory results. Despite the benefits of this study, scaffolding-related falls have not been thoroughly explored. Also, our experience shows that more superior hardware speed, performance, and capacity are generally needed to develop a VR model. The National Research Council (NRC) Committee on the R&D on Virtual Reality [Durlach and Mavor 1995] recommended the use of Onyx Reality Engine2 (Onyx RE2) graphics workstation for building VR models. This has been the reason for us to acquire and use Onyx RE2 in our ongoing project--Construction Operations using Virtual Reality (COVR). The three-year project, partially funded by the National Science Foundation (NSF) for undergraduate education, calls for the development of construction operation models. This machine was also used to develop the virtual environment of SAVR.

CHAPTER III
SAVR CHARACTERISTICS

3.1 Introduction

In this chapter, the specific characteristics of SAVR training scenarios are introduced, which include the description of the training environment, the specific causes of falls, and the interactive training scenarios in SAVR.

3.2 The Training Environment

The training environment in SAVR is a virtual construction site for constructing reinforced concrete walls. The concrete walls are assumed to be constructed and formed using plate girder form, which supports the form scaffolding platform. Both the plate girder and form scaffolding are manufactured by the Economy Forms Co. (EFCO). The form scaffolding provides the working platform to inspect the concrete form and re-bars, and to pour the concrete. Figure 3-1 shows the scaffolding components in a virtual environment.

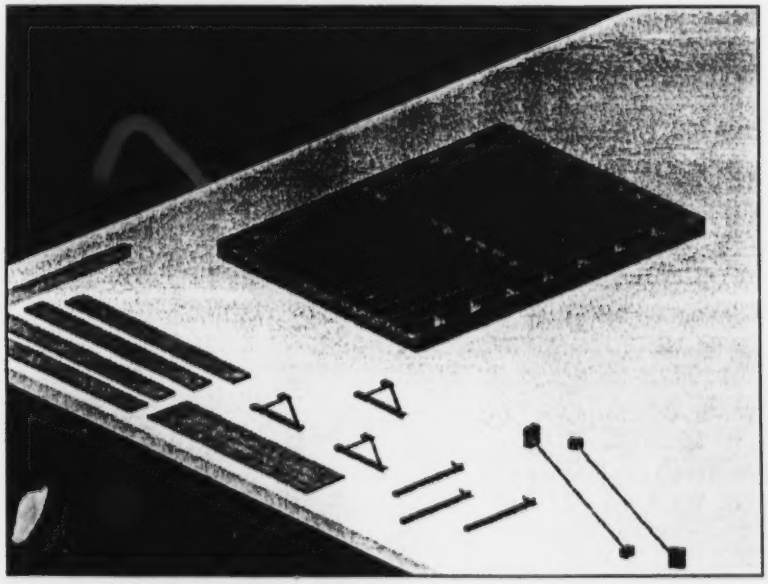

Figure 3-1. Scaffolding components as shown in a virtual environment.

BEST COPY AVAILABLE

The scene includes several roads and green areas to provide the feeling of depth (3D space) to trainees while being immersed in the environment. Since these surroundings are only used as a background, they were modeled using less detail than the scaffolding components, which were modeled carefully to present all the required detail for training workers, such as the position of bolts and nuts connecting the scaffolding components. These details enable the representation of fall causes in scenarios using SAVR.

3.3 Fall Causes in SAVR Scenarios

As mentioned in Section 2.3 fall causes can be of four major types: enabling, triggering, support-related, and conditioning, which can be related to the worker and/or the supporting platform including safety devices. In SAVR we emphasized the importance of the support-related and conditioning causes through the visual representation of several causes, such as missing and defective components. Causes that are related to each scaffolding component were then grouped into three major problem types: component problems, connection problems, and other miscellaneous problems.

3.3.1 Component Problems

Each of these types of causes was related to a problem with a single scaffolding component which affects the integrity of the platform and may lead to a fall accident. These problems include the following:

(1) A damaged (bent) bracket.

(2) A Missing plank.

(3) A cracked plank.

(4) A bent post to support the guardrail system.

(5) A Missing top rail, mid rail, or toeboard from the guardrail system.

3.3.2 Connection Problems

Concerning connection problems, causes were related to the bolts and nuts connecting the scaffolding components, which may be missing from the connection between any two components. These problems considered connections between:

(1) The formwork and the bracket.

(2) The bracket and the plank.

(3) The bracket and the post.

(4) The post and the top rail, mid rail, or the toeboard.

3.3.3 Miscellaneous Problems

Additional causes of falls include:

(1) Excessive spacing between the brackets (exceeding the maximum allowable).

(2) Insufficient plank overlap.

(3) Overloading planks with excessive construction load.

The visual representation of the causes of falls in these problem groups was created in interactive training modes as described below.

3.4 Interactive Training Scenarios in SAVR

The training scenarios in SAVR were based on two major tasks in using the form scaffolding platform: erecting the platform and inspecting an existing platform. These two tasks were implemented in SAVR using two modules: the erection and inspection modules. In the erection module, trainees participate in the virtual environment by installing each of the platform components through the proper installation sequence, and the appropriate position and connection. An interface panel including erection instructions is provided to install users which components need to be installed. Trainees can install each component using a special interface device--a cyberglove--while immersed in the environment using the HMD.

In the inspection module, the scaffolding section is already erected and includes problems explained earlier in Section 3.3. Trainees are allowed to explore the scaffolding and identify each problem correctly. An interface panel is provided for them first to choose the problem type, find its location, and select it using the mouse. Once both the problem type and location are selected correctly, trainees earn a score which is also included in the interface panel. In addition, the program shows the correction of that problem.

An example of the inspection scenario is selecting from the interface panel the connection problem between the bracket and formwork. Trainees have to find this particular problem to earn their score. If they select the wrong location, their score will not change. The inspection module is also supported by sound files that describe each problem.

While both modules are supported with sound files to simplify their use, a help module is developed separately to teach users how to interface with SAVR. The help module mainly uses graphics and sound files, which are expected to be more effective than traditional help functions that only use text files. The help module is also provided with an interface panel to select among several help topics. The interface panels in all modules can be switched between two modes: visible and invisible. While the visible mode enables users to enter their selections, the invisible one provides a more immersive environment which only contain form scaffolding components in the virtual site.

CHAPTER IV
HARDWARE AND SOFTWARE FOR SAVR

4.1 Hardware

The hardware in this project included two platforms: a PC and a graphics workstation. The PC platform was a Pentium P5-60 with 16 MB of RAM, a keyboard, a mouse, and a SVGA monitor. The use of a PC here coincides with the availability of our software that was acquired for use in a PC environment.

A graphics workstation was also used to develop SAVR's virtual environment. The main reason for using a graphics workstation was our future plans to expand the scope of SAVR to include more training scenarios for several scaffolding platforms, which will require sophisticated hardware. The hardware used was an Onyx Reality Engine 2 (Onyx RE2), a high performance graphics system from Silicon Graphics Inc. (SGI).

The Onyx RE2 has many special features as a graphics system. Among these features is an advanced rendering capability. The system currently has one Reality Engine2 which can render 1.6 million triangles per second [SGI 1994]. The texture mapping memory is 4MB, which can be extended up to 16MB. Additionally, the system provides 128 MB of RAM, which can be expanded up to 16 GB. These are only a few of Onyx RE2's features, it has many other advanced graphics and computational capabilities. The Onyx RE2 configuration is much more sophisticated compared to any other available PC configuration. In fact, Onyx RE2 was designated as one of the best platforms for developing VR models by the National Research Council National Research Council Committee on Virtual Reality Research and Development [Durlach and Mavor 1995].

Two special interface peripherals are employed for users to interface with SAVR. Both peripherals, which allow users to have immersive interaction, are connected to the Onyx RE2 through a tracker Polhemus 3D Space FASTRAK which tracks a user position and orientation. The first peripheral is an Eyegen3 head-mounted display (HMD) from Virtual Research Co. and the second peripheral is a cyberglove from Virtual Technologies Inc.

4.2 Software

In developing SAVR, the following software was used:

(1) World Tool Kit (WTK) Development System Version 2.10 for SGI and Version 2.02 for Windows from Sense 8 Co.,

(2) 3D Studio Version 3.00 from Autodesk Co.,

(3) Microsoft Visual Basic Version 3.0, and

(4) Utility programs including Windows Screen Printer "SnagIt 2.0.07", and LVIEW for Windows Version 3.1, WinSock File Transfer Protocol (WS_FTP), and tri_quad from WTK Users' Group.

4.2.1 World Tool Kit Development System

World Tool Kit (WTK) was the main software for developing SAVR. Two WTK versions were used in this study. The SGI version was used in the Onyx RE2 platform and the Windows version was used for the PC platform. In both versions, the package included a library of C functions [WTK 1995]. The library functions in both versions were almost the same; however, there were some compatibility limitations, particularly in hardware dependent functions. In both versions, the functions were grouped into different classes providing a tool for developing interactive virtual worlds using the object-oriented programming feature. The important features in WTK are introduced in detail in Chapter V.

4.2.2 3D Studio

The 3D Studio package was used in the PC platform to construct the 3D graphical objects that are used in SAVR, such as the 3D model of the scaffolding. It was also used to create several image files which were used for texture mapping. The use of 3D Studio to construct 3D graphical objects is described in a previous study by Barsoum [1995]. New modeling techniques in constructing SAVR models are discussed in Sub-Section 5.1.1.

4.2.3 Microsoft Visual Basic

The Visual Basic (VB) package was used in the PC platform to construct the graphics for the interface panels in SAVR modules. Each panel, including its buttons, was developed to give the panel a professional appearance --similar to the appearance of the Windows-based interface panels.

4.2.4 Utility Programs

One utility program used was the Windows Screen Printer "SnagIt 2.0.07". This program was used in the PC platform to capture the images of the interface panels created by VB. Using SnagIt, the captured images were stored in "bitmap" file format--the only available format in this program. The LVIEW for Windows package was used to convert

the image format from "bitmap (BMP)" to "targa (TGA)" and scale the image size using the standard size for "targa" images.

The WinSoc File Transfer Protocol was used to transfer the image and 3D data files from the PC platform to the Onyx RE2. Afterwards, image files were converted from the "targa" format to the "RGB" format, which was the only readable format in WTK for the SGI version. This conversion was done using the "fromtarga" SGI utility program which converts "targa" files to "RGB" files. The 3D data files which were created using the 3D Studio package were also converted using the utility program tri_quad, which takes a 3D file in 3D Studio file format (3DS) format or the WTK Neutral File format (NFF) and outputs a 3D file in the NFF format [McClarnon, 1995]. The program is a public shareware for WTK users and was downloaded from the WTK Users' Group site (SIC-WTK).

CHAPTER V

IMPORTANT FEATURES IN WORLD TOOL KIT

5.1 Application development Using World Tool Kit

The basic operations to develop an interactive virtual world using WTK are shown in Figure 5-1. These operations include the construction of the 3D graphical objects, the preparation of texture maps (images), and writing the code using the WTK C library functions [WTK 1995]. The construction of the 3D objects can be done using WTK or other external graphics packages, such as AutoCAD or 3D Studio. To give the 3D objects a more realistic representation texture maps are usually applied to their surfaces. These maps can be photographed or video captured from real objects and converted to 2D images, or they can be created using paint programs. In order to complete the construction of the interactive virtual world, the WTK C library functions are used to represent the appearance and behavior of the 3D objects and to allow users to interact with these objects in the virtual world.

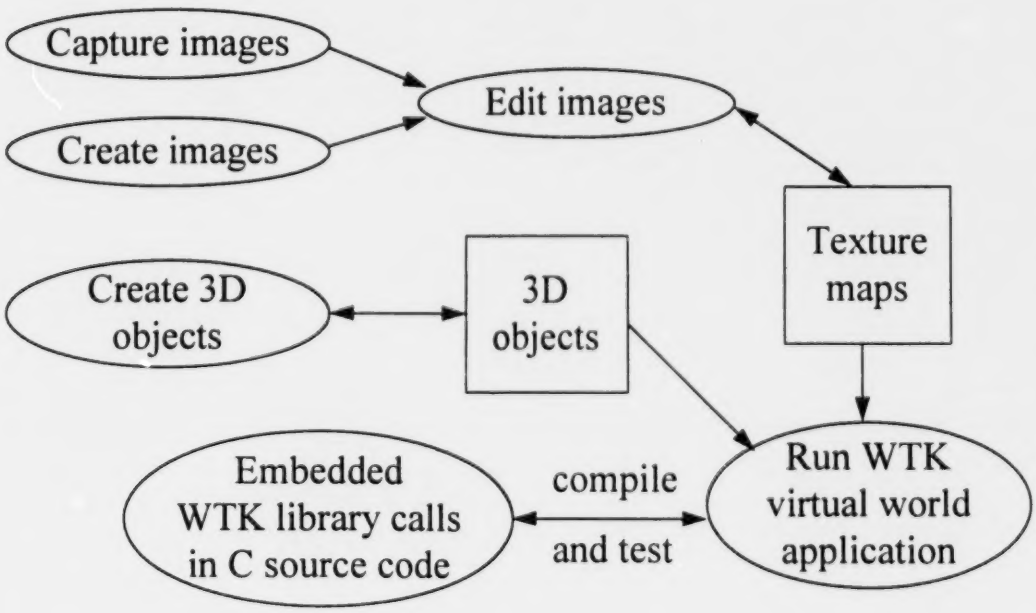

Figure 5-1. WTK application development process [WTK 1995].

A WTK virtual world is presented to the users as a computer-generated 3D space that includes all objects and is called the universe. An application can use different universes; however, only one universe can be active at a time. Graphical objects can be added to or removed from a universe. Several universes are used in the SAVR virtual world and all of them contain several 3D graphical objects. The following Sub-sections 5.1.1 and 5.1.2 describe the construction of the 3D graphical objects and the preparation of texture images that were applied to their surfaces.

5.1.1 Graphical Objects Construction

In constructing graphical objects for a virtual environment, the rendering capability of the hardware is of a great concern. While these objects can be detailed to look similar to real-life objects, to render objects having complex detail during the simulation requires extensive hardware computations. In such a case, the number of frames rendered per second (fps) may be reduced and a real-time rendering may not be achieved. Accordingly, hardware limitation is the governing factor in deciding the degree of complexity for the objects.

In constructing the graphical objects for SAVR, we considered the limitations of the platform. We maintained all the required detail for training, such as the detail of several scaffolding components, and eliminated the unnecessary detail, such as the detail of the surroundings. This procedure furnishes users with a real-time interaction on the workstation platform and a clear representation of the key details for training.

According to their degree of detail, the graphical objects in SAVR were one of two types: simple or complex objects. Simple graphical objects were those objects constructed using few surfaces (polygons) for graphical presentation. The most simple graphical object was constructed using a single polygon. For example, each control panel included in SAVR was constructed using a single polygon. However, a texture image was applied to the panel. Single polygon objects were constructed using separate WTK NFF files. Each object was defined as a quad (quadrilateral) shape by using four vertices that define the X, Y, and Z coordinates of each of its corners. The name of the object and its accompanied texture were also specified inside the NFF file.

Besides single polygon objects, other simple objects were constructed using a low number of polygons, such as the surrounding green areas and the wooden components of the scaffolding. For example, the graphical object represented the plank component was constructed using 3D Studio, which was easier to use than the WTK NFF for constructing 3D objects. The 3D Studio represents object faces using triangular polygon, which required the use of 12 polygons to represent the plank. The *tri_quad* program was used to convert the 3D Studio triangular representation (Figure 5-2) into a quad representation (Figure 5-3), which reduced the number of polygons by 50%. The final quad representation was stored using the NFF format, in which a texture image of wood was applied to the plank surfaces to provide a more realistic appearance than only colored surfaces. Generally, the construction of simple graphical objects was an easy task

compared to the construction of the complex graphical objects, such as the steel components of the scaffolding.

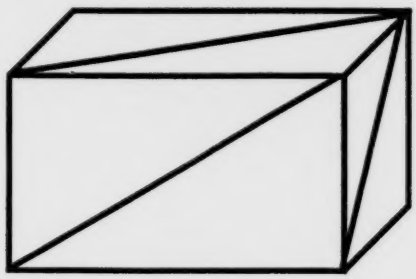

Figure 5-2. A 3D Studio triangular representation of faces

Figure 5-3. A quad representation of faces

All complex objects in SAVR were constructed using 3D Studio. The method of constructing complex objects was presented in an earlier study by Barsoum [1995]. An example of these objects in SAVR is the graphical object represented the bracket (Figure 5-4). This object was assembled from steel angles and each angle included the required holes to connect the bracket with other scaffolding components, such as the formwork or the post. Representing these details in each steel member increased the required number of polygons to present the object. To illustrate, the number of polygons representing one bracket was 3565. Complex objects were also converted to the NFF format using the *tri_quad* program to reduce the number of their polygons by 50%. Since these objects consist of a higher number of polygons compared with simple objects, texture images were not applied to these polygons--only color was used to represent their material.

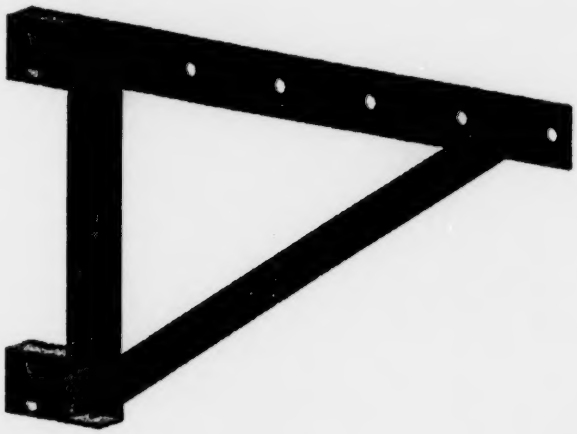

Figure 5-4. A graphical model of a steel bracket component modeled after EFCO

5.1.2 Texture Images Construction

The application of texture maps on several object surfaces in SAVR provides these objects with a more realistic appearance than the application of one color. For example, the wooden texture applied to plank surfaces represents the plank surface better than using one color only. Images were also used to construct the user-interface panels in SAVR which increases its user-friendliness.

Texture images were constructed using several file formats including: the bitmap, targa, and RGB formats. While the bitmap format was used as a transition format, the targa and RGB formats were the two final storage formats for each image included in SAVR. This was because the WTK for PC could only process the targa format and the WTK for SGI could only process the RGB format, which required converting all other image formats to the targa and RGB formats. To illustrate, the construction of the image (Figure 5-5) applied to the surface of the control panel in the inspection module was done in the following manner:

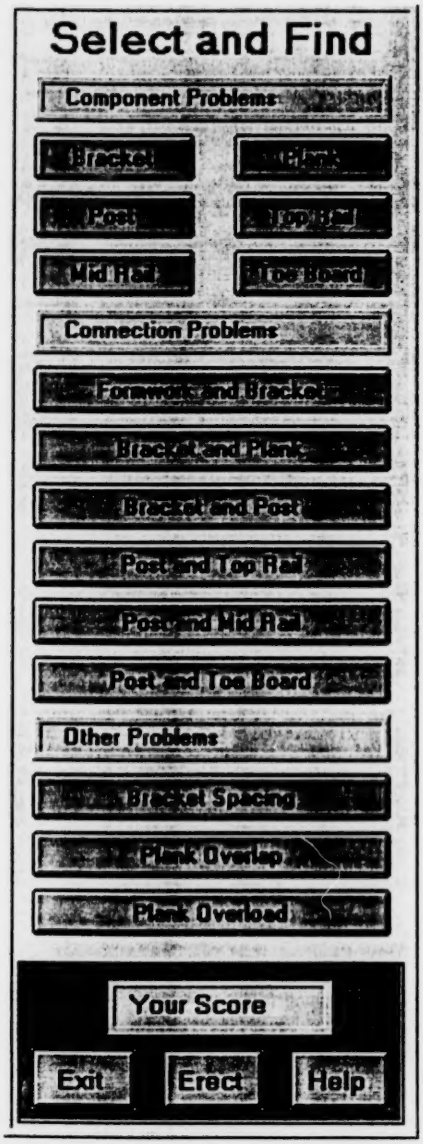

Figure 5-5. The image applied to the surface of the inspection interface panel

(1) Using the Visual Basic (VB) package to construct the panel frame and buttons, which also included assigning a label to describe the function of each button.

(2) Using the Windows Screen Printer, *SnagIt*, to capture the image of the inspection panel from the VB window. The captured image was stored using the bitmap format, which was the only available format for capturing an image using *SnagIt*.

BEST COPY AVAILABLE

(3) Using the *LVIEW* program to convert the bitmap format to the targa format. The dimensions of the resulting image were 233 pixels wide by 624 pixels high; however, WTK requires that the dimensions be in multiples of four, such as 320 pixels and 480 pixels, to enable processing of an image. Accordingly, the image was re-scaled using *LVIEW* to a width of 240 pixels and a height of 640 pixels. At this stage the image was ready to be used by the WTK in the PC platform, in which all images were stored using SAVR's image library.

(4) In order to use the image in the Onyx RE2 platform, the *WinSoc FTP* was used to transfer the image to the platform. This image was then converted to the RGB format using the SGI "fromtarga" utility. Finally, the image was also stored in SAVR's image library in the Onyx RE2 platform.

The above procedure was followed to construct all texture images in SAVR. Although using this procedure consumed a substantial amount of time, it provides SAVR's interface panels with a professional appearance.

The construction of the graphical objects and texture images was an essential task for SAVR development; however, these objects had no dynamic or interactive feature; they were required as objects to be incorporated into the virtual world. Various dynamic behavior and interactive characteristics were assigned to several of these objects using the WTK classes and functions, which is introduced in Section 5.2. Further, Section 5.3 describes the major tasks performed for an application developed using WTK. Both sections provide the necessary background information to clarify the use of WTK to develop SAVR.

5.2 The World Tool Kit Development System

As mentioned earlier in Sub-Section 4.2.1, both WTK versions (for PC and SGI) included hardware dependent functions which can only be used for a particular platform. In order to have a portable code for both platforms, the hardware dependency limitations were overruled by implementing conditional statements to check the hardware components and, consequently, choosing the appropriate functions to execute.

In both versions, the functions were grouped into several classes using object-oriented naming conventions, which point to these classes [WTK 1995]. For example, the function *WTobject_ delete* belongs to a class of graphical objects and is used to delete an object. All functions in this class start with the convention of *WTobject_* and are followed by the function name. WTK classes include the *universe*, *graphical objects*, *viewpoints*, *lights*, *sensors*, and other classes. In the following sections, a few WTK classes are presented.

5.2.1 The Universe Class

In WTK terminology, the simulated environment is referred to as the *universe*, which is a computer-generated 3D space that contains all WTK simulated objects, such as

graphical, view point, light, and sensor objects. In an application, several universes can be simulated; however, only one universe can be simulated at a time. The universe class contains high level functions that can interface with any simulated objects at any time during program execution.

All of these functions in the universe start with the convention *WTuniverse_* and are followed by the function name. These functions are sub-grouped according to their use, for example, there were subgroups for universe construction and destruction, loading and saving, simulation management, viewpoints, geometrical properties, graphical objects and polygons, intersection testing, and performance statistics [WTK 1995]. Several of the universe class functions are introduced throughout this chapter according to their use in SAVR's development.

5.2.2 The Graphical Objects Class

To provide a computer-user interface, VR applications mainly rely on the graphical objects. This class includes the functions that are needed to interface with the state of a graphical object. Similar to the universe class, the functions are sub-grouped according to their use. For example, the subgroup for object management includes the functions that are used to load a graphical object from a data file into the universe, delete an object from the universe, copy the object, or save an instance of the graphical object to a new data file. Other subgroups include object rendering, position, orientation, intersections, tasks, and hierarchies [WTK 1995].

5.2.3 The Sensors Class

Sensors are the input devices that allow users to be immersed in the simulated environment and interact with its graphical objects. There are different types of sensors, such as a mouse, a spaceball, or a head tracker. More than one sensor can be used in an application and be attached to graphical, light, or viewpoint objects [WTK 1995]. For example, a mouse sensor can be attached to the viewpoint to represent the viewer position and orientation in the universe at any instance.

This class includes the function subgroups which interface with the common sensor types. All of these functions start with the convention *WTsensor_* and are followed by the function name. Two sensor types were used in SAVR: the standard mouse and the Polhemus FASTRAK. The standard mouse was used to manipulate the graphical objects and to define the viewer's position and orientation. The Polhemus FASTRAK was connected to the HMD to detect the viewer orientation, and the cyberglove to detect its position, orientation, and pre-defined user-actions, such as specific gestures.

5.2.4 The Viewpoints Class

The viewpoint parameters define how the simulated environment is projected to the display device [WTK 1995]. The simulated environment can be projected to several

windows using several viewpoints; however, only one viewpoint can be used for each projection to a window. Even though only one viewpoint can be used for any window projection, in the case of using only one window, different projections can be generated when switching to different viewpoints.

The WTK SGI viewpoints class facilitates generating a monoscopic or a stereoscopic view. When using the monoscopic view, both eyes can see the same rendered view. On the other hand, a stereoscopic view generates two different views: a left and a right eye rendered view. While the stereo view can be more realistic than the monoscopic view, it is not supported on PC platforms. Since we expect that in the future, SAVR may be implemented on PCs, the monoscopic view was used.

In general, the viewpoint class includes the functions that are needed for viewpoint management. These functions start with the convention *WTviewpoint_* and are followed by the function name.

5.3 The Major Tasks in An Application Using WTK Functions

The Major tasks in an application using WTK functions include:

(1) Initializing the simulated environment.

(2) Specifying the interactive scenarios during the simulation.

(3) Entering the simulation loop and performing the simulation.

5.3.1 Initializing the Simulated Environment

Initializing the simulated environment involves creating the universe, including its objects, such as graphical, light, and sensor objects. However, these different object types may be changed, deleted, or replaced during the simulation. Generally, initializing the simulated environment includes:

(1) Initializing the display size and position.

(2) Loading the required 3D graphical objects to build the virtual world. Graphical objects may be loaded in the forms of either stationary or dynamic objects, according to their use in the application (stationary and dynamic objects are described in more detail in Section 5.4).

(3) Assigning the required task to each dynamic object.

(4) Initializing the position and orientation of the viewpoint, which represents the initial position and orientation of the viewer (user).

(5) Loading the lights data file.

(6) Creating the sensors which enable the user to interact with the environment.

After initializing the simulated environment, the interactive scenarios during the simulation must be specified.

5.3.2 Specifying the Interactive Scenarios During the Simulation

The interactive scenarios during the simulation are pre-defined by the developer and depend on the nature of the application. These scenarios are always based on defining special events of interest (actions) and the accompanying required reactions to each action. An action may be a single event or a group of events and similarly, an interaction may require performing a single task or a group of tasks.

The implementation of these scenarios in WTK applications is done using a function called the *universe action function*. This function controls the flow of the simulation scenario and is executed at every simulation loop to detect any of the specified actions and consequently, execute the reactions. For example, the erection module allows users to walk through the site indifferent directions, such as moving up down, right, and left using the cyberglove. Each direction can be selected using a unique hand gesture. This simple scenario required pre-defining the movement in each direction and its accompanied gesture (conditions), detecting gestures (actions), and matching both conditions and actions to check if both of them are true.

The universe action function facilitates the definition of any number of events to control the simulation scenario; however, programmers must consider optimizing its performance to minimize the execution time, otherwise every simulation loop may take a relatively long time, which may reduce the frame rate of the application. This is explained in more detail below.

5.3.3 Starting the Simulation Loop and Performing the Simulation

The simulation loop is considered the heart of any WTK application [WTK 1995]. Once the simulation loop is entered, the universe is brought to life, including all of its objects. The simulation loop is entered when the function *WTuniverse_go* is called, which is typically called only once during the program execution. The loop is iterated continuously until the function *WTuniverse_stop* is called, which is also typically called only once. The simulation loop (Figure 5-6) performs the following tasks:

(1) Reading all sensors, which includes reading the data from all active input devices, such as a mouse, a head tracker, or a cyberglove. The keyboard input is also read at this stage; however, WTK handles the keyboard differently from sensor objects.

(2) Calling the universe action function and checking sensor input with the pre-defined events in the function to execute the required actions, if any. The universe action function usually includes at least one event that causes the calling of *WTuniverse_stop* in its pre-defined events, which ends the simulation.

(3) Updating objects according to sensor inputs. The universe objects need to be updated relative to the viewer's new position and orientation after every action. In addition, sensors may also be used to move or rotate a particular dynamic object, which requires updating that object.

(4) Allowing dynamic objects to perform their tasks. For example, if an object's task is to spin around a particular axis, this task will be performed in every simulation loop.

(5) Rendering the universe, which is the last task in every iteration and which results in one frame.

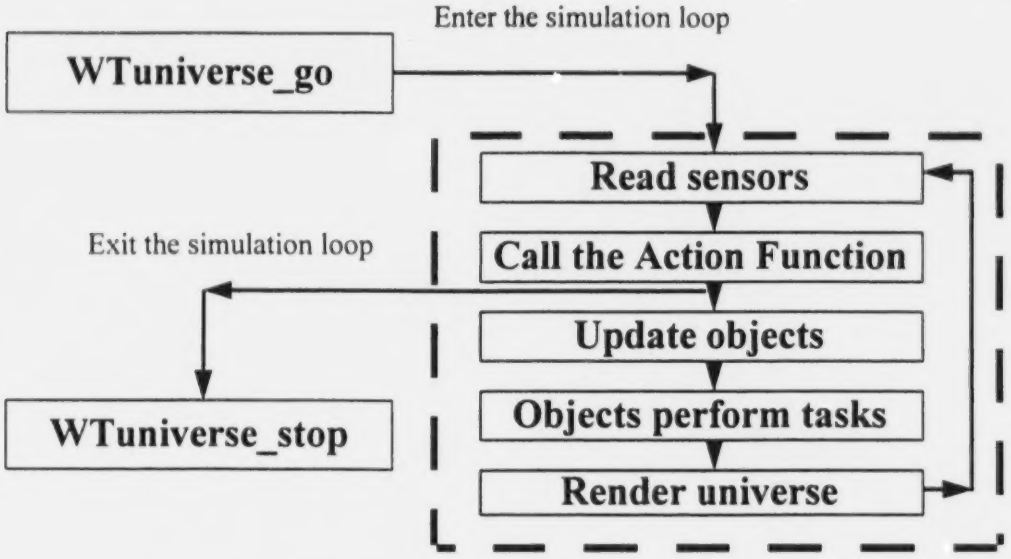

Figure 5-6. The default simulation loop [WTK 1995].

The order of tasks 1 to 5 is the WTK default order for the simulation loop; however, the order of tasks 2, 3, and 4 can be changed, if required, as long as Tasks 1 and 5 remain in the same order.

Obviously, the faster the simulation loop, the higher the frame rate of an application. While both reading the sensors and rendering the universe mainly depend on the hardware, the total time required for one iteration can be minimized through optimizing the universe action function, object tasks, and the number and complexity of the graphical objects that are rendered every iteration.

In a WTK simulated environment, a graphical object can be one of two types: stationary or dynamic. The following section describes both types.

5.4 Types of Simulated Graphical Objects

The type of a graphical object during the simulation depends on how the object is loaded into the universe [WTK 1995]. If an object does not have to perform any task

during the simulation, it should be loaded as a stationary object. Examples of stationary objects in SAVR are the surroundings, such as the roads and green areas which were considered background objects which require no changes during the simulation. Conversely, the scaffolding objects in SAVR change during the simulation and perform specific tasks, such as checking intersections with other objects to detect collisions every simulation loop. Such objects, which were required to do tasks, were loaded in the form of dynamic objects. There were two functions for loading objects, one for each object type.

The first function was *WTuniverse_load* and was used to load a stationary graphical object into the universe using a graphics data file, such as an NFF or a 3DS. A universe may either contain no stationary object or only one stationary object as a background. However, if multiple stationary objects are needed, these objects can be grouped in one graphics data file and loaded as one stationary object. The position, orientation, and geometry of a stationary object cannot be changed during the simulation; however, its color and texture can be changed. A stationary object cannot have a behavior or be attached to a sensor.

The second function was *WTobject_new,* which was used to load a dynamic graphical object. Many dynamic objects can be included in a universe. WTK provides several functions to manage any dynamic object within the simulation. This includes changing the position, orientation, color, texture, and geometry of the object.

A dynamic object can be loaded or deleted during the simulation. Creating a new object during the simulation may affect the frame rate, particularly when loading a complex graphical object. In such a case, the object can be created before starting the simulation loop and be added to the universe when required, which is done by using the functions *WTobject_remove* and *WTobject_add.* If an object will not be required in the simulation anymore, the function *WTobject_delete* is used to delete that object. The behavior of a dynamic object during the simulation can be implemented by using the object's task function.

5.5 The Object Task Function

The object task function is a user-defined function which specifies the behavior of a dynamic object during the simulation [WTK 1995]. Similar to the action function, the object task function is called to execute the behavior of the dynamic objects every simulation loop. Dynamic objects may perform different tasks during the simulation; however, each object can perform only one task. A task can be assigned to a dynamic object by passing the names of the object and its task function to the function *WTobject_settask*. An object task can be updated during the simulation by re-passing the new name of the task function to the same function. Furthermore, an object task can be deleted when passing the object name to the function *WTobject_deletetask*.

The task function allows developers to control the behavior of each dynamic object during the simulation. This function can be as simple as spinning an object around a specific axis, which can be implemented using a single instruction. On the other hand, it

can be very complex and include sub-function calls inside its body. Since the task function is executed during every simulation loop, programmers should consider not defining tasks that may require a relatively long time to execute, which may reduce the frame rate.

The above sections introduced background information for using WTK. Based on this background, the development of SAVR is discussed next.

CHAPTER VI
THE CONSTRUCTION OF THE SAVR PROGRAM

6.1 Development Approach

In developing SAVR, a six-step problem solving approach following Adams et al. [1988] was reviewed: (1) defining the problem, (2) designing the solution, (3) refining the solution, (4) considering a testing strategy, (5) coding, testing, and debugging the program, and (6) documenting the program. This approach provided the general conceptual framework for problem solving using a computer which was adapted for SAVR development. In this section, each of the six steps is described in detail.

6.1.1 Defining the Problem

From a programming perspective, the problem in developing SAVR was to produce an application for training construction workers to erect and inspect a form scaffolding structure in order to maintain a safe scaffolding platform. The previous general description was divided into two sub-problems: erection and inspection. Each sub-problem in turn was divided into several smaller sub-problems, and so on.

One of the most important issues in SAVR's problem definition was defining the input and output scenarios to solve the problem. The input was defined as the user's action(s) in the virtual environment while erecting or inspecting the scaffolding. For example, to erect the bracket component, the input was defined as the user actions to bring the component into the environment, to pick it up using the cyberglove, and to move it to its correct position. The output in this case was the visual simulation of the bracket movement to the new position. Thus, the solution was designed based on the problem definition, the possible input from users, and the desired output scenarios.

6.1.2 Designing the Solution

The solution to the above problem was to design an algorithm to develop the simulated environment, which facilitates the training process. This global picture of the solution was divided into two main environments; one for erection and the other for inspection. However, a third environment was also needed to provide the user with the necessary help instructions for using the program. Moreover, a start environment was also required to allow the user to choose among these environments.

Each environment was implemented in the SAVR program using a separate module, which included only its related scenarios. For example, the inspection module included only the inspection scenarios in which defective and missing scaffolding components were presented, and the user was required to detect each of them. This modular design was essential for developing a well-structured application. After the main modules were defined, which represents the major tasks for the application (Figure 6-1), the solution was refined into more detailed tasks.

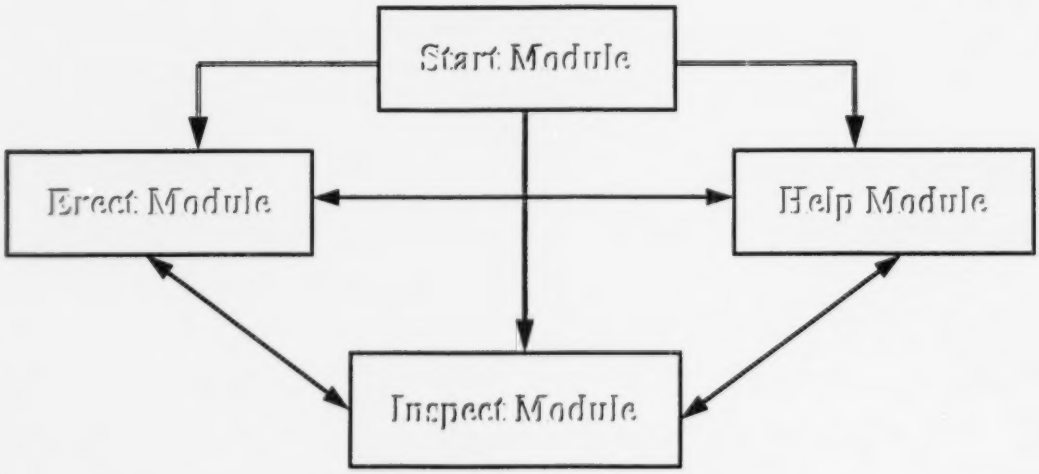

Figure 6-1. SAVR's main modules.

6.1.3 Refining the Solution

The tasks in every module of SAVR were outlined using a *pseudocode solution*, which is a representation of the solution procedure for that module using simple statements without any special programming syntax [Adams et al. 1988]. For example, the pseudocode solution for the inspection module was primarily represented through the following steps:

(1) Loading the scaffolding components to be inspected.

(2) Loading other graphical objects required in this module, such as the user-interface panel and the surroundings.

(3) Positioning the viewer in the inspection environment using an initial position and orientation.

(4) Starting the simulation loop and reading the sensors to detect the user's action(s):

- If the user identifies a problem correctly, call the scoring function to increase the score and display the solution to this problem.

- If the user chooses to exit the simulation, end the program execution.
- If the user chooses the erect module, call the erect module.
- If the user chooses the help module, call the help module.
- If the user did not choose any of the above actions, redo Step (4) (the user decision is still undecided, loop until a decision is made).

These Steps were further refined using the *step-wise refinement* technique, which calls for dividing a sub-task into more detailed sub-tasks. For example, Step (1), which called for loading the scaffolding components for inspection, was refined to specify the name of each scaffolding component to be loaded.

The step-wise refinement of the pseudocode solution enabled us to represent the logic using simple steps, which simplified the coding of the application using the C language. Before starting the coding stage, a very important task to consider was a testing strategy to debug the code.

6.1.4 Considering a Testing Strategy

The visually oriented output from SAVR provides a simple tool (visualization) to test the program performance during the simulation; however, it does not provide the cause(s) of an error (bug) if one occurs. Accordingly, we considered a strategy to check if the program correctly detects user inputs and appropriately responds to them. Our strategy was to break the code down into small portions and test the performance of each portion during the program execution. This was done by implementing print statements to show which portion of the code is executed and the data manipulated in that portion, such as changing the position of a graphical object or replacing it during the simulation. The use of these print statements helped to limit errors in the small portions of the code, which in turn simplified the debugging process.

6.1.5 Coding, Testing, and Debugging the Program

As mentioned earlier (Sub-Section 6.1.3), coding the SAVR program was simplified by using step-wise refinement to represent the application algorithm. The code was mainly constructed using the C programming functions included in the WTK library. However, several user-defined functions were also constructed using the C standard functions combined with WTK functions, such as the scoring function, which was constructed using a combination of WTK and standard C functions to update the user score while using the inspection module. During the coding stage the program was tested frequently to detect any bug, find its location and cause, and correct the code.

6.1.6 Documenting the Program

SAVR was documented during its development to include every modification. This practice not only simplified the final documentation, but also simplified the testing

and debugging operations. Even though the documentation was a time-consuming task, it would have been impractical to maintain and expand the code (over 1300 lines) without good documentation practice.

The development steps in this section were overlapped and/or repeated several times to optimize the application performance and to suit other programming considerations, which are described next.

6.2 Programming Considerations

Other specific programming considerations were of major concern in SAVR's development. These include the following: (1) the programming environment, (2) the structure of the program, (3) the flow of the program, (4) the Graphical-User Interface (GUI), and (5) the potential for expanding the application.

6.2.1 The Programming Environment

The C language, in which the code has to be compiled to object files, was the programming environment used for SAVR development. These object files must be linked to other object files from different libraries in order to produce the application executable file. In order to generate the SAVR executable file, two libraries were required, the standard C library and the WTK library. A WTK *makefile,* which was modified for SAVR development, was used to specify the compilation and linking instructions, and to point to the locations of the required libraries.

Since SAVR uses graphical object and texture data files during its execution, an environment variable was set to point to the location of these files, as was instructed in the WTK manual [WTK 1995]. Both environment variables are set to point to the locations of the libraries, including SAVR's models and textures respectively, as follows:

WTMODELS=/home-directory/savr-models

WTIMAGES=/home-directory/savr-images

Other data files such as lights and sounds are stored in SAVR's main directory, */home-directory/savr,* in which no environment variable needed to be set.

6.2.2 The Structure of the Program

SAVR was structured according to the general rules for C programming and was assembled from the WTK functions and other user-defined functions. Similar to all C programs, SAVR has a *main* function, which is executed first to call other functions. Each of these functions returns to the *main* function after performing its task(s). The last call at the end of the *main* function is to terminate the application.

In the SAVR structure, some functions were developed to perform a specific task for a particular module, such as the *ScoringTask* function, which is only called by the

inspection module. Other functions were designed to be used with more than one module, such as the *AlignTask* function, which is called by each module in SAVR to re-align the user-interface panel relative to the viewer position. Besides these functions, the SAVR structure included four major function groups. Each of these groups consisted of two functions that were related to one, and only one, of the SAVR modules. The first function in each group was designed to initialize the simulation in a particular module and the second function was to control the simulation in that particular module. These four groups are as follows:

(1) *LoadMainScreen* and *MainScreenActions,* which are used for the start module,

(2) *HelpModule* and *HelpActions,* which are used for the help module,

(3) *ErectModule* and *ErectActions*, which are used for the erection module, and

(4) *InspectModule* and *InspectActions,* which are used for the inspection module.

All of these functions, which were built using WTK and other C functions, are only a sampling of SAVR's user-defined functions. Using each of the above function groups is essential for clarifying the flow of the SAVR program, which is introduced below.

6.2.3 The Flow of the Program

The flow of the SAVR program (Figure 6-2) is mainly controlled by the *main* function. This function initializes the universe and calls the *LoadMainScreen* function in order to load the graphical objects for the start module environment. After loading these objects, the control returns to *main*, which assigns the *StartActions* function for the simulation and subsequently calls the *WTuniverse_go* to enter the start module simulation loop (the simulation loop is described in Sub-Section 5.3.3). Once the loop is entered, the flow of the program is temporarily controlled by the *StartActions* function.

The *StartActions* contains four pre-defined conditions to allow users to select among the erect, inspect, or help module, or to terminate the program. When any of the previous conditions is true, the *StartActions* calls the function *WTuniverse_stop* to end the simulation of this module after assigning a numeric value to a global variable called *ChooseModule*. This value, which is used by *main* to detect the user's choice, can be one of four integer values: 1, 2, 3, or 10. The first three values are associated with choosing the help, erect, and inspect modules respectively (values from 4 to 9 are reserved values for new modules in SAVR that may be needed for its future expansion). The fourth defined value (10) is a special one for program termination--when this value is detected by the *main*, the function *WTuniverse_delete* is called to terminate the program. Similar to *StartActions*, all other action functions in SAVR have pre-defined conditions for assigning different values to *ChooseModule*, which are also used by *main* in the same manner.

In this flow design, when the user chooses any of the SAVR modules, the current universe is deleted and a new universe is loaded, which only contains the graphical

objects pertaining to the chosen module. In addition, every action function includes only the scenarios pertaining to the chosen module and thereby, no unnecessary checks are performed for actions related to other modules during the simulation. Even though switching modules requires a few seconds before loading a new universe, the simulation loop within any universe was optimized, which minimized rendering time. While considering the flow design for SAVR, its graphical-user interface (GUI) was also taken into account to provide a user-friendly interface.

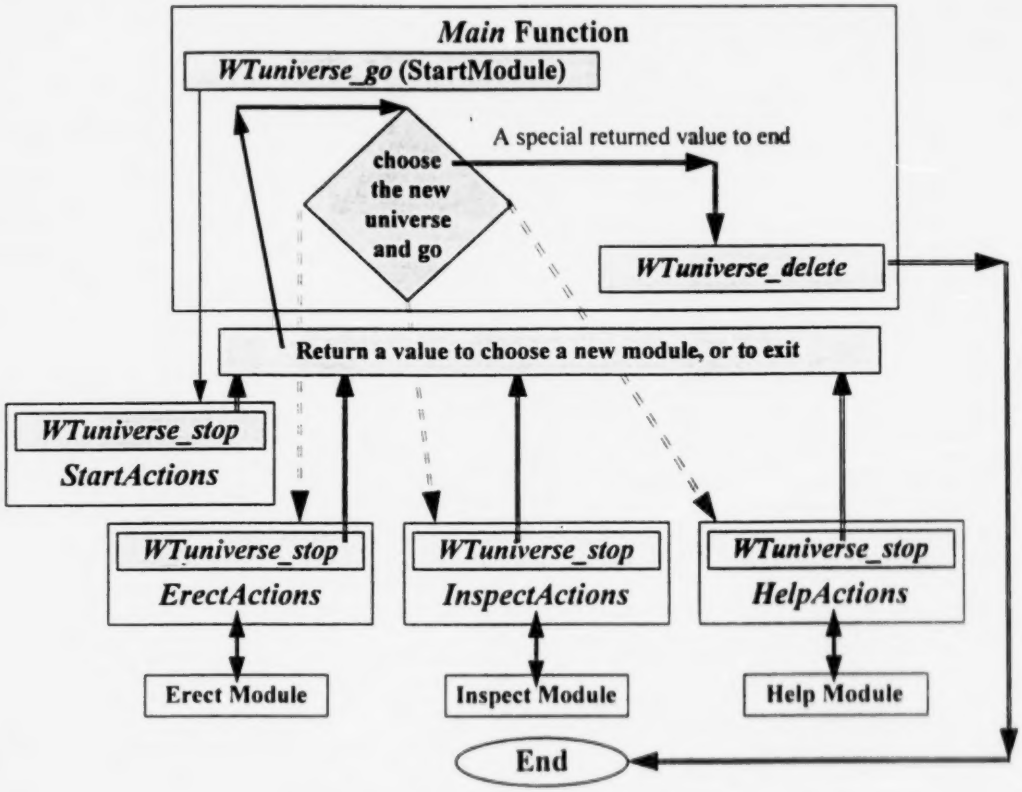

Figure 6-2. The flow of the SAVR program.

6.2.4 The Graphical-User Interface

SAVR was developed for users with different levels of computer expertise, such as novice engineers, students, or construction workers. The GUI was designed to provide a user-friendly interface which requires a minimum amount of skill to use. For example, users only need to click a mouse button to interact with the graphical objects in a virtual environment. Furthermore, users can pick an object using a cyberglove, which does not require them to type any command. The keyboard option was also included, however, when using the HMD, the keyboard is not likely to be practical.

BEST COPY AVAILABLE

The GUI in SAVR provides users with an interface panel for each module. Every module panel includes a unique set of interface buttons used to perform the specific tasks in that particular module. The implementation of these panels was done using several 2D graphical objects, which were textured to define the function of each button. These control panels simplify the training process; however, they required more effort to design the button functions and code the application. The WTK classes do not include any special function to create interface panels or buttons.

In addition to the control panel, users can use the help module for further explanation. The help module control panel includes several help buttons to play audio files in order to explain how to use the program. The appearance of SAVR's GUI is shown through snap-shots of different modules. Figure 6-3 shows a snap-shot of SAVR's start module, including its control panel. Figure 6-4 shows the erection module display. The control panel in this module includes several buttons to load different scaffolding components in order to erect the scaffolding platform, such as the hand rail, bracket, and plank components. Figure 6-5 shows a snap-shot of the inspection module and its interface panel. It also shows a missing plank from the platform, which has to be installed to maintain a safe platform. The programming techniques for developing a user-friendly GUI are discussed in more detail in Section 6.4.

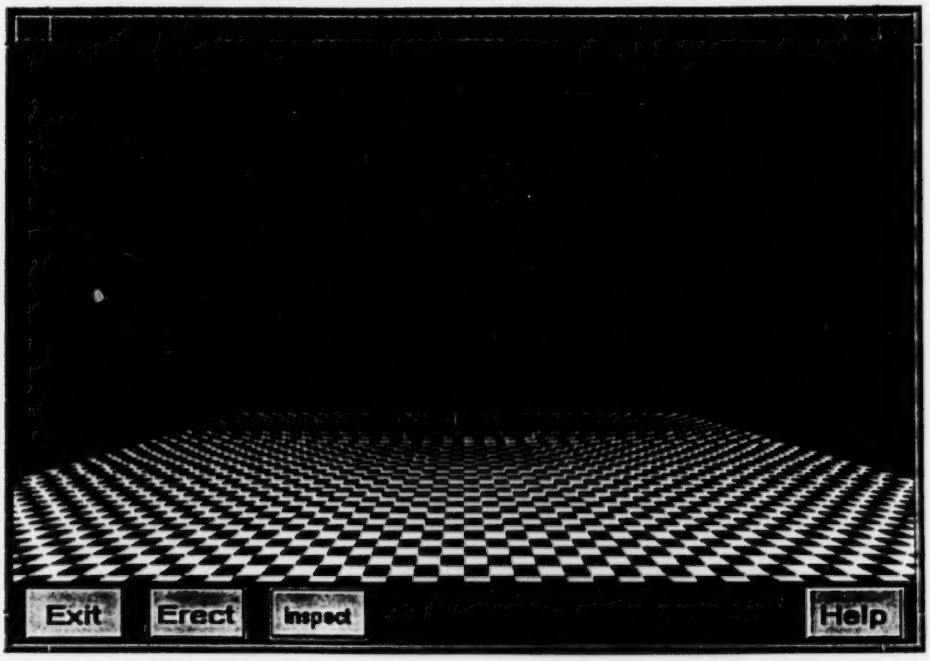

Figure 6-3. A snap-shot from SAVR's start module.

BEST COPY AVAILABLE

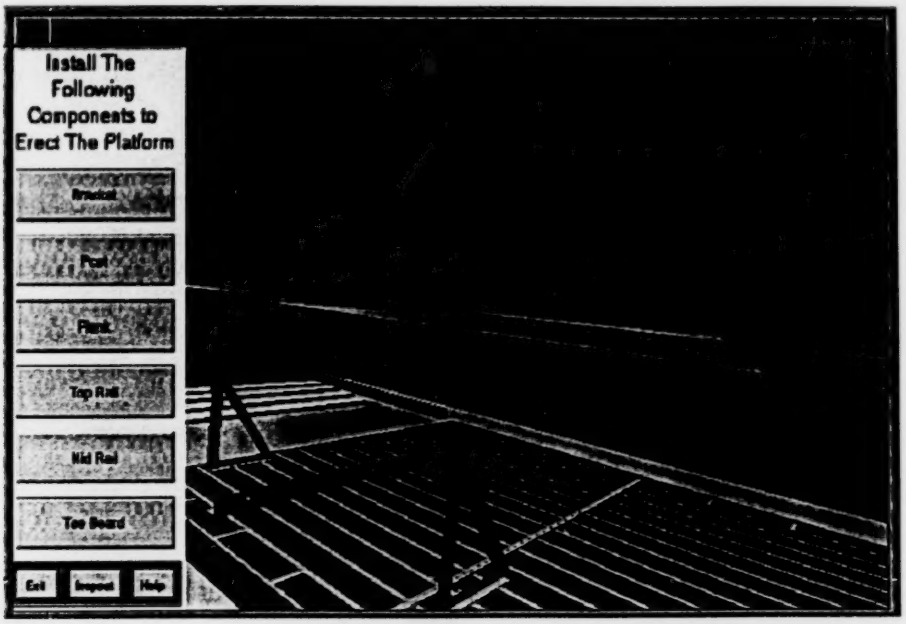

Figure 6-4. A snap-shot from SAVR's erection module.

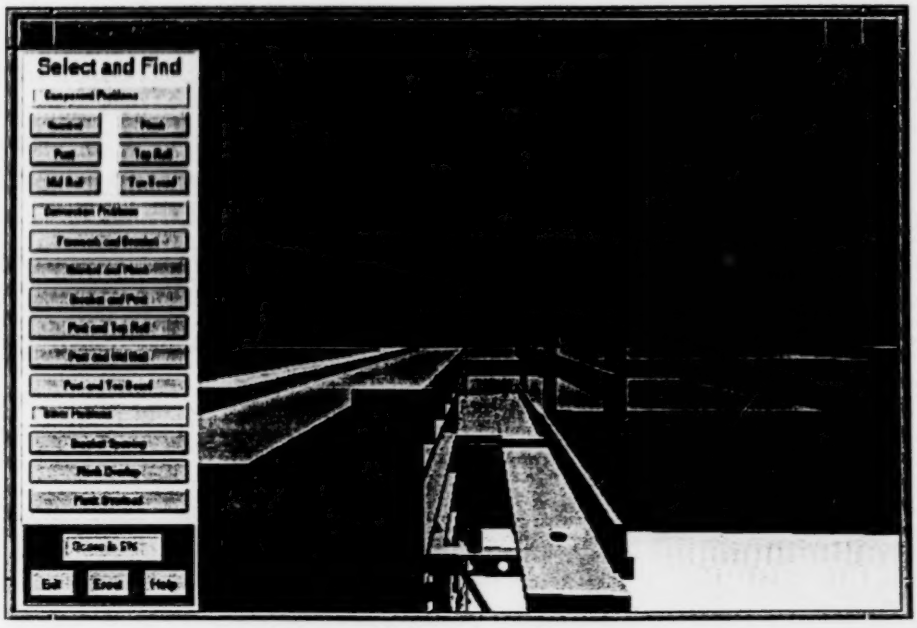

Figure 6-5. A snap-shot from SAVR's inspection module.

BEST COPY AVAILABLE

6.2.5 The Potential for Expanding the Application

The implementation of several modules and action functions in SAVR allows the application to be expanded to include other types of scaffoldings and training scenarios. In addition, a new module and its new action function can be incorporated with minor changes in the code. Furthermore, such new modules can be developed and checked separately and be added to the application with minor effort.

6.3 The Design and Implementation of SAVR's Interactive Scenarios

The interactive scenarios can be described as sets of pre-defined actions and interactions. The set of pre-defined actions includes the pre-defined user inputs during the simulation. These predefined inputs can be detected through reading the data from the input devices and sensors at every simulation loop. The set of pre-defined interactions includes the program response to each of the pre-defined actions. The interactive scenarios of SAVR's modules are implemented within the action function controlling the simulation in that particular module. For example, the interactive scenarios for the erection module are implemented within the *ErectActions* function, which controls the interactive scenarios in the erection module. The design and implementation of the interactive scenarios in the erection module are described below.

6.3.1 The Design and Implementation of the Erection Interactive Scenarios

In order to design the interactive scenarios for erecting the form scaffolding in SAVR's virtual environment, the erection process in the real environment was carefully considered and accordingly, the major erection tasks were identified. These tasks are performed in the following order:

(1) Connecting the brackets to the plate girder. The plate girder at this stage is horizontally positioned on the ground.

(2) Connecting the posts to the brackets.

(3) Connecting the toeboards to the posts.

(4) Installing the planks on the brackets.

(5) Moving all of the above components to the position where the concrete wall is to be formed. At this stage the plate girder is vertically positioned to form one side of the concrete wall.

(6) Installing the diagonal braces to support the plate girder in the vertical position.

(7) Connecting the intermediate rail to the posts.

(8) Connecting the top rail to posts.

Since these tasks are sequentially performed in the real surroundings, the interactive scenarios in the virtual environment must be performed in the same order. These interactive scenarios are implemented within the *ErectActions* function.

At the beginning of the simulation, the *ErectActions* function allows the user to perform Task (1). It allows the user to pick one of the three brackets, which must first be installed on the plate girder. The *ErectActions* function checks every simulation loop to determine which bracket was chosen. When the user picks one of the brackets, the *ErectActions* executes a sub-function called *ErectBrackets*, which allows the user to install the selected bracket on the plate girder. This is done through checking the intersection of this bracket and any of the pre-defined appropriate positions for bracket installation. When the intersection is detected, the program frees this bracket from the virtual hand and installs it in the appropriate position. The user can then pick up the second and third brackets, one at a time, and install them using the same procedure. Once all three brackets are installed, the *ErectActions* function disables the *ErectBrackets* sub-function and allows the user to perform Task (2) in the erection procedure.

Similar to Task (1), Task (2) involves picking one of three posts and installing it at the appropriate location. The *ErectActions* function executes a sub-function named *ErectPosts* to facilitate the installation process using the same technique described in Task (1). For the erection Tasks (3) through (8), the *ErectActions* function executes a particular sub-function for each Task. Each of the previous Tasks allows the user to pick and install one of the scaffolding components.

The picking and installation are done in the erection module using the cyberglove. SAVR users are enabled to use the cyberglove and pre-define specific hand gestures to pick scaffolding components and walk through the erection environment. These gestures are stored using a data file named *SAVRgestures*. SAVR pre-defined gestures include: *Forward*, *Backward*, *Up*, *Down*, *Right*, *Left*, *Stop*, and *Pick* gestures. During the simulation the *ErectActions* function checks if the user's hand gesture matches any of the pre-defined gestures to facilitate the interaction. For example, if the user gesture matches the *Forward* pre-defined gesture, *ErectActions* moves the user forward through the environment. While the use of the cyberglove in SAVR was an important feature to develop a user-friendly graphical-user interface (GUI), other techniques were also used to enhance SAVR's user-friendliness.

6.4 Techniques for Developing a User-Friendly Graphical-User Interface

SAVR's GUI provides users with a control panel at every module to choose among several options. It also includes a scoring system to evaluate user performance during their training for erection and/or inspection. The implementation of SAVR's control panels and scoring system is discussed below in Sub-Sections 6.3.1 and 6.3.2 respectively.

6.4.1 Design and Implementation of SAVR's Control Panels

As mentioned earlier, WTK does not provide a special class for creating control panels or buttons; however, they were created in a form of 2D graphical objects and implemented using a user-defined task to perform their required tasks during the

simulation. The control panels in SAVR modules were of two types. The first type was a single 2D graphical object which used one texture map applied on its face to represent all of the buttons and their functions. This type was used when many buttons were required for user interface, such as in the case of the inspection module, in which 18 buttons were represented. The employment of a single object optimizes the use of the texture memory, since only one texture map was needed to represent all the buttons. However, this type required more programming effort to detect each of these buttons.

The user-defined tasks in SAVR's control panels involved two major tasks. The first was developing an algorithm for aligning a panel, including its objects, to a fixed projection on the WTK window during the simulation. The second task was to activate the buttons when chosen by the user.

6.4.1.1 Fixing the Projection of SAVR's Control Panels

The graphical objects in SAVR, including those of the control panel, are projected to the screen relative to the viewer position and orientation during rendering time. Since SAVR allows the user to walk through the simulated environment, the user position and orientation (the universe viewpoint) is expected to change at any time during the simulation, which in turn changes the projection of the graphical objects. In order to overcome this projection change of the control panel and maintain the same projection during the simulation, a WTK object hierarchies linkage and the function *AlignTask* were used. The logic used for fixing the projection of SAVR's control panels is presented using the flow chart in Figure 6-6.

In using the WTK object hierarchy, an object can be attached to another whose movement is identical to the first [WTK 1995]. In such a case, the first object is called the parent object, while the second is called the child object. The parent object can have more than one child. Object hierarchies can be of different levels; however, in SAVR only one level was used, with one or several child objects attached to one parent object.

The parent object in SAVR was a dummy object which was only used to control the movement of its children. This dummy object was a graphical object to be loaded into the universe at the same position and orientation of the universe viewpoint to represent the viewer at the beginning of any simulation loop. The size of the dummy object was designed so that the object is always invisible to the user. The child objects were the graphical object(s) which presented a control panel at a particular module. For example, when activating the start module, the dummy object was loaded first at the same position and orientation of the universe viewpoint. Then, the graphical objects of the control panel were loaded and each one was attached to the dummy object. These attachments caused panel objects to follow the dummy object during the simulation. The following two lines of code were used to attach the background *panel* and the *Exit* button objects to the dummy before entering the simulation loop:

```
WTobject_attach(DummyObj, Panel);
WTobject_attach(DummyObj, ExitButton);
```

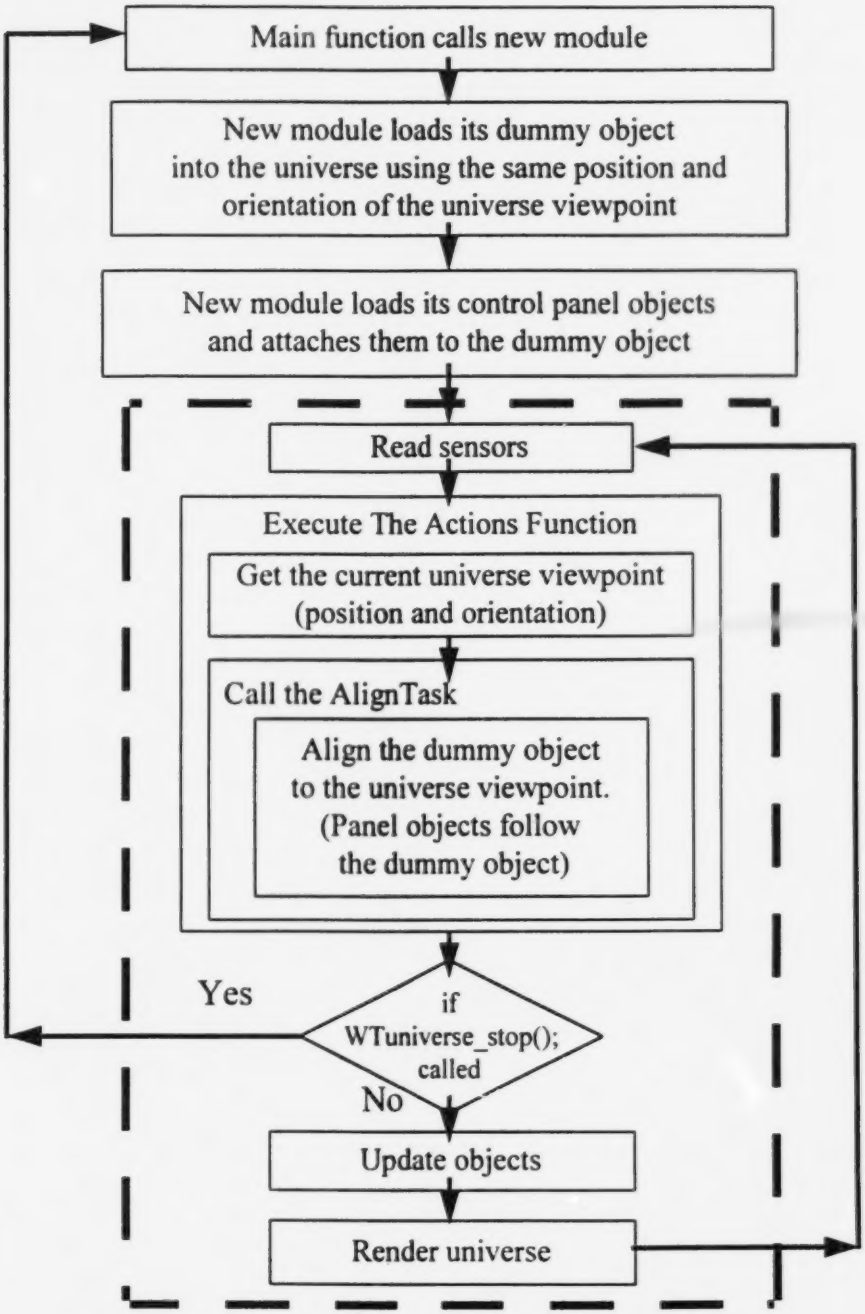
Main function calls new module
New module loads its dummy object into the universe using the same position and orientation of the universe viewpoint
New module loads its control panel objects and attaches them to the dummy object
Read sensors
Execute The Actions Function
Get the current universe viewpoint (position and orientation)
Call the AlignTask
Align the dummy object to the universe viewpoint. (Panel objects follow the dummy object)
Yes
if WTuniverse_stop(); called
No
Update objects
Render universe

Figure 6-6. Fixing the projection of SAVR's control panels.

Since the position and orientation of the universe viewpoint changes during the simulation, the position and orientation of the dummy, followed by its children, must also

change the same values. The function *AlignTask* performs these changes in SAVR by getting the values of the position and orientation of the universe viewpoint and assigning the same values to the dummy object.

AlignTask is a user-defined function which was implemented in SAVR as a general (global) function to be called by all SAVR's action functions. *AlignTask* is called every simulation loop, regardless of the active module. The function must be called after reading the sensors and before rendering the universe, in order to update the position and orientation of the dummy object. Once the dummy object is updated its children are also updated, which maintains the same projection of the control panel within the active WTK window.

6.4.1.2 Activating SAVR's Interface Buttons

In SAVR, an interface button can be selected by placing the mouse cursor on that button and clicking the middle mouse button at the same time. The reason for using the middle mouse button for selection, rather than the left or the right one, was that the middle button does not cause any change in the viewer's position or orientation, and thereby the environment projection (the scene) does not change.

In order to activate the interface buttons, the program must first detect which button was clicked, and consequently perform the button function. In SAVR, two methods were used for detection through accessing the data of the mouse during the simulation loop. The first method was used when each button was presented using a unique graphical object, such as the start module buttons, in which four graphical objects were used to represent the buttons. The second method was used when only one graphical object represented all the buttons, such as the case of the inspection module where one graphical object represented all buttons through the use of texture.

The first detection method was coded based on the following steps, which are executed by the action function every simulation loop:

(1) Get the data from the mouse sensor and store it in a temporary location.

(2) If the middle mouse button is not pressed go back to Step 1, otherwise:

- Pick the nearest graphical object, on which the mouse cursor overlays any of its visible polygons.
- Get the name of the selected object
- If no object was picked, redo Steps 1 and 2

When an object is picked, its name is stored in a temporary string variable called *ButtonName* (A string variable is a variable which can be used to store an array of characters). The action function compares the value of this variable with its pre-defined conditions, which are also implemented using different strings. For example, the following conditional statement is used to check if the Exit button is selected by comparing two strings

if (strcmp(ButtonName , "b-exit") == 0)

The "*b-exit*" string is the pre-defined string in the action function and the *ButtonName* is the other string for comparison. Once a user clicks the *Exit* button, the value of the variable *ButtonName* is updated. The updated value is the name of the object that was clicked, "*b-exit*", which is read from the NFF file containing the object. Accordingly, the conditional statement becomes true and the instructions inside its body, which represent the function of the *Exit* button, are executed. The pre-defined events of choosing any of the interface buttons in SAVR are also defined using string values. An important step in using this method for activating a button is to check if the name of the object in the NFF file is identical to the pre-defined string, otherwise, the button would not be activated.

In the second detection method, we used the same procedure described above to detect whether the user clicked the interface panel; however, since only one object included all the buttons, an algorithm was used to identify the selected button. This algorithm calculates the position of the mouse sensor relative to the interface panel when the middle button is clicked. This relative position can be compared with the relative position of each of the interface buttons to check if the mouse sensor has overlaid any of them, which would identify the chosen button. The values used to calculate the mouse position relative to the WTK window are shown in Figure 6-7.

The values *XO* and *YO* specify the x and y positions of the lower left corner of the WTK window relative to the computer display, which may be changed at any time during the simulation if the user moves or maximizes the window. The values *W* and *H* identify the width and height of the WTK window, which may also be changed if the window is maximized or re-sized. The *XO, YO, W,* and *H* values can be accessed using the WTK functions *WTwindow_getposition* as shown in the following statement:

WTwindow_getposition(WTuniverse_getwindows(), &xo, &yo, &W, &H);

The values *Xm* and *Ym* indicate the mouse x and y positions relative to the computer screen, which depend on the position of the mouse cursor when clicking the interface panel. Since all of the previous values can be changed during the simulation, they must be updated before being used in any calculation, which is only performed when the middle mouse button is clicked while overlaying the interface panel. The following two equations are used to obtain the *Xr* and *Yr* values, which present the x and y positions of the mouse cursor relative to the WTK window:

$$Xr = (Xm-Xo)/W$$
$$Yr = (Ym-Yo)/H$$

Since the *Xr* and *Yr* values are x and y ratios (dimensionless), they do not depend on the size or location of the WTK window. Each button in the interface panel can be identified using similar ratios to present its x and y coordinates relative to the WTK window. This required identifying the lower left corner (*Xmin, Ymin*) and the upper right corner ratios (*Xmax, Ymax*) for each interface button to identify its range. These ratios were obtained using the same method for calculating *Xr* and *Yr*; however they are fixed ratios for each button and do not change during the simulation because the projection of the panel is already fixed relative to the WTK window. During the simulation, the values

Xr and *Yr* are compared with the X and Y ranges of each button to check which button is clicked, if any.

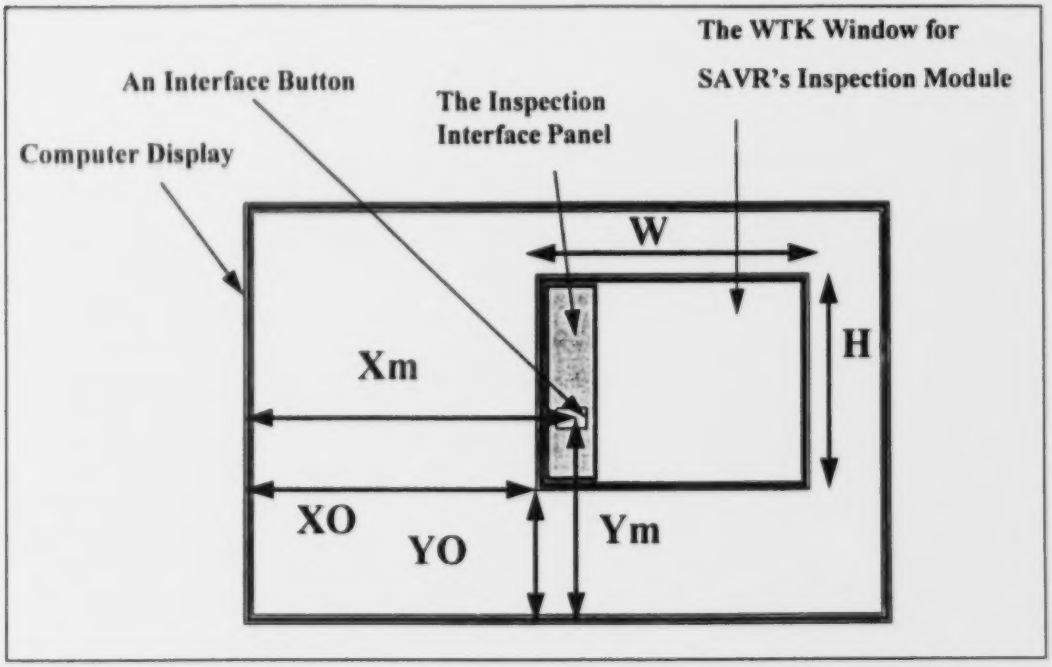

Figure 6-7. The values used to calculate the mouse position relative to the WTK Window.

6.4.2 Design and Implementation of SAVR's Scoring System

The purpose for having a scoring system in SAVR was to evaluate users' ability to identify hazardous conditions and eliminate them from the scaffolding platform. These hazardous conditions were predetermined in the program and can be visually identified when using SAVR's inspection module. The removal of each hazard requires the user to click the hazard and identify its type. For example, the removal of a hazard due to the existence of a broken guardrail requires the user to find this broken guard rail, choose it using the mouse, and choose the *Top Rail* button from the *Component Problems* group. This case is presented in the *InspectActions* function using a conditional statement to match the selected component and the problem definition. If the conditional statement becomes true during the simulation, which means that the user has found the hazard and correctly classified it, the user earns a partial score.

The score ranges from 0% to 100%. It is implemented using 5% increments. The score is displayed inside a box within the inspection interface panel using a texture map.

Since only one texture map is used for this panel, this texture is replaced with another every time the user earns a partial score. Textures are stored in the directory using file names that indicate their contents. For example, the texture files *Panel-00* and *Panel-25* include the textures of (0%) and (25%) scores respectively.

The code for implementing the scoring system consists of two separate parts. The first part, which is implemented in the *InspectActions* function, is to check the removal of the hazardous conditions. The second part is a user-defined function, *ScoreTask*, which when called by *InspectActions*, checks the last score and increments it to update the user's score. In order to check the last score during the simulation, *ScoreTask* checks the name of the texture placed on the interface panel. This scoring system adds another dimension to the SAVR training system, through which trainees can improve their skill by repeatedly running the sessions.

6.5 Model Transformation from Onyx to PC

Two WTK versions were used in this study. The SGI version was employed for the Onyx RE2 platform and the Windows version was used for the PC platform. In both versions, the package included a library of C functions. The library functions in both versions were nearly identical; they were grouped into different classes providing a tool for developing interactive virtual worlds using the object-oriented programming feature. Both WTK versions also included hardware independent functions. An efficient way to achieve platform portability was to plan ahead, so transformation of SAVR models should be done relatively easily at any point in time. Hence, transformation was planned particularly at the early stages of developing SAVR models.

The plan to transform SAVR from one platform to another relied on the use of the hardware independent functions to develop most of the application code. The SAVR model was first developed in the Unix environment using the WTK SGI version. While most functions in the SAVR structure were selected to be executable on the PC platform, few UNIX-based functions were used to optimize the rendering process on the Onyx RE2. To create a transformable model to the PC platform, the use of UNIX-based hardware dependent functions were avoided in the program. This was a simple modification for using the code in the PC platform otherwise a considerable amount of time would have been required for the transformation. Another simple modification was made by changing the format of textured images used in Onyx RE2 from the RGB format, to the TGA format that can be used in the PC platform. This was done using an image utility program, namely ImageMagick. Note that other utility programs, such as PhotoShop and Adobe could also be used for this purpose. Then, all SAVR's data files (3D objects and textured images) were transferred to a PC using a file transfer program, File Transfer Protocol (FTP). The source code files were recompiled on the PC resulting in a working PC version of the model. Figures 6-8 and 6-9 show examples of images of the form scaffolding and control panel captured from the PC environment. The PC we used was a Pentium 60 with 16 MB RAM without a graphic accelerator. We expect the model performance to improve considerably when used in an enhanced PC environment.

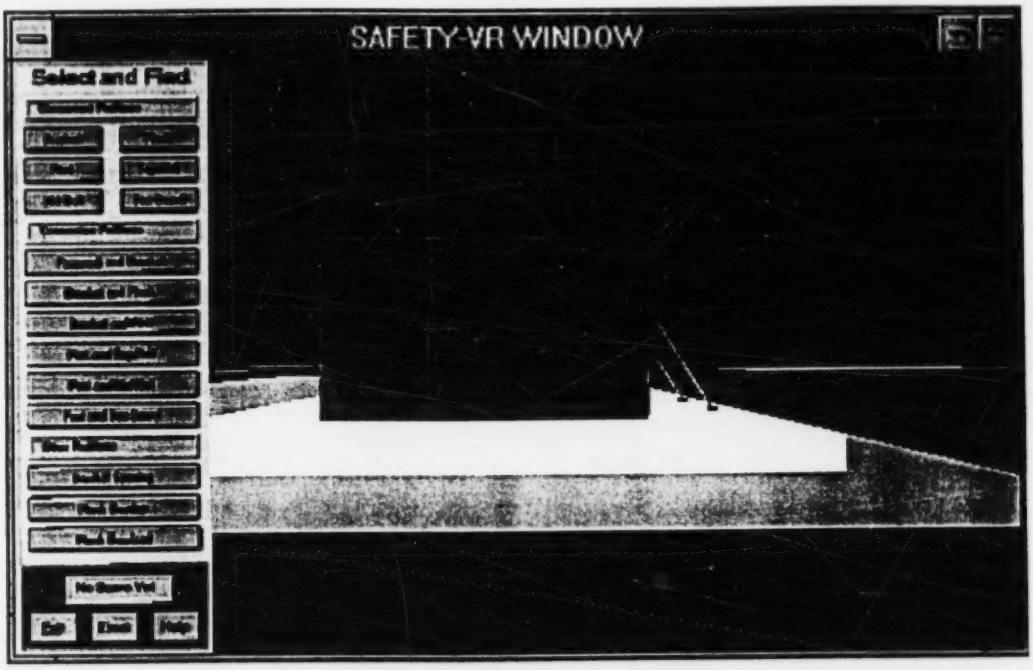

Figure 6-8. An image captured from the PC environment.

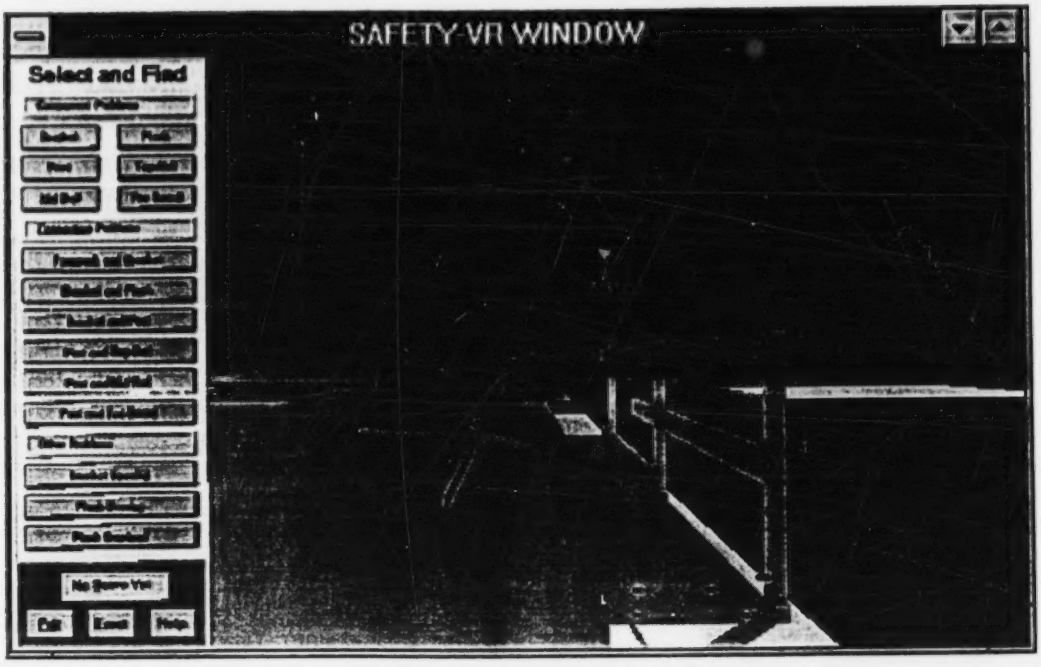

Figure 6-9. An image captured from the PC environment.

BEST COPY AVAILABLE

CHAPTER VII
EVALUATION AND TESTING

7.1 Introduction

To ensure that SAVR achieves its intended purposes, we evaluated, updated, and upgraded the system continuously throughout the developmental stages. This continuous evaluation has made major modifications less likely, particularly at the later stages of the development process. Furthermore, we tested and evaluated the system to detect weaknesses, such as features that did not perform their intended function, logic of operations, conditions that were not previously considered by the system, and additional features required as suggested by the experts. Finally, evaluation and testing of the system by independent experts was conducted upon completion.

7.2 SAVR Evaluation and Testing

The evaluation and testing of SAVR by the developers was performed in two stages: first, during the knowledge acquisition stages, a confirmation is required to use VR technology and to determine whether VR was the most appropriate technology to handle the given problem as opposed to other more traditional training tools. This confirmation came from an evolutionary process in performing research in construction safety using simulation tools. Results of our previous studies suggest that interactive simulation tools alone (e.g., expert system, multi-media, etc.) may not be sufficient to impress the danger involved in construction operations on trainees. Hence, an immersive environment that can only be achieved through the use of VR technology is a natural step to enhance our previous system. Such a confirmation was supported by the experts during the knowledge acquisition process performed in our previous studies as well as in this study.

In the second stage, evaluation and testing on the four environments (erection, inspection, start, and help) were continuously performed to ascertain the correctness and completeness of the erection and inspection modules. In the erection module, the sequence of erecting the form scaffolding components in a safe procedure is of particular importance. In the inspection module, adequacy in establishing visually common problems in form scaffolding components is essential. Also, logic in establishing and solving such problems is checked. The start and help environments are evaluated for their professionality and user-friendliness.

7.3 Formal Evaluation

A formal testing and evaluation was conducted for the inspection module by eight experts from construction industry and safety institutions. The evaluation criteria as well as the questions used are shown in Figure 7-1. The following expressions were used as evaluation criteria for the graphical representation and system performance:

- **Excellent:** no improvement is required
- **Very good:** little or no improvement is required
- **Good:** some improvement is required but no major mistakes are detected.
- **Fair:** some problems are detected but performance is satisfactory.
- **Poor:** major changes are required.
- **Very poor:** redevelopment is required.

7.4 Results of Formal Evaluation

To see if our model is realistic, the graphical representation of the model was assessed by experts on two issues, i.e., detail (dimension and size) and appearance (closeness to real-life appearance). Figure 7-2 shows their assessment regarding the detail of the scaffolding components. The average of the experts' assessment suggests that the components detail is very good. The appearance of the model was judged to be slightly better than very good, as can be seen in Figure 7-3.

SAVR's performance was evaluated based on the following issues: user interface, problem/solution representation, applicability for training, and overall performance. The majority of experts judged the user interface as between very good and excellent (Figure 7-4). The result seems to suggest that SAVR can be used comfortably by the end-users without difficulty. Experts assessed the representation of the problems and solutions designed for SAVR as a little better than good (Figure 7-5). This "rather low" assessment (compared with other issues) from the experts is to be expected since problems and solutions were designed for entry level construction employees. Figure 7-6 shows experts' opinion on SAVR's applicability as a training tool which indicates an average assessment of between good and very good. Finally the overall performance of SAVR was judged by the experts as close to very good, as illustrated in Figure 7-7.

Besides their assessments on SAVR, the experts were also asked to comment on their interest in using SAVR, the benefits of the system, potential impact, critical needs for safety training, and the role of academic community in addressing their needs in improving safety (Figure 7-1). Six out of the eight experts indicate that they are "very interested" in using SAVR for training their construction workers. One suggests that "It's probably the 'training choice' of the future."

With regards to the potential benefits of SAVR as the experts see it, the following is a selection of their comments:

- "The SAVR would allow the individual to explore at their own level."
- "An easier way to build and detect hazards without exposing any employees to any hazards. With the SAVR program employees can be trained in classroom anytime throughout the work shift also it can be a refresher course."

- "The formwork & scaffolding was modeled after my company's equipment. My company could use this in training my sales and field service people."
- "Graphical image of true to life problems encountered on construction sites."

Some comments on the potential impact of SAVR follow:

- "In the future I could see the systems such as the SAVR being the basis for most of construction safety and health training. The system obviously has far reaching implication."
- "This is a good way to show mock-ups of typical problems in construction and other industries."
- "The program could serve as "competent person" training and perhaps qualify those workers who 'pass' the tests as 'OSHA-qualified' competent persons. SAVR can provide a great way to learn about the safety without being physically exposed to real risks."
- "When used in applications such as building construction, ditching, pipe laying, etc. I see it as a tremendously valuable tool."

On the critical needs for safety training in the next five years, their comments include:

- "Fall protection, electrical, trenching."
- "Trenching, scaffolds, general industry 'safety of workplaces,' tunneling in the Columbus area, roofing."
- "Roofing contractors and fall protection, trenches – protect workers from cave-ins, etc."
- "(1) Falls, (2) electrical, (3) struck by hazards, (4) caught by hazards, (5) waste handling and clean up (nuclear), (6) emergency response to spills, explosions, fires, (7) violence in work place."
- "You name it, we need it! Starting with fall protection and scaffolding, you're on the right track. New kinds of training--through computers etc.—is the way to go... the traditional lecture classes are not as effective for people who by nature, are "hands on" (like people in construction)."

On the role of academic community in addressing the above critical needs, the experts have these comments:

- "Use of modern technology to aid in the training of construction workers."
- "Keep up the R&D work, with assistance from "real life" construction companies who can work with you on practical knowledge vs. theory. I'd like to see this program expanded to include other types of scaffolds."
- Educating students prior to them joining the work force will be good."
- "(1) Engineering and marketing need to know the hazards of the products and systems they will design and sell. (2) Safety of humans should be designed into all systems—this must be taught. (3) The proper assembly, use and limitation of systems should be clearly incorporated into all system design. . . . Where possible all systems should be standardized to reduce the need for new unexpected problems or hazards."

SAVR EVALUATION FORM

Evaluator: _____

Position: _____ **No. of years of experience:** _____

Affiliation: _____

Address: _____

Phone: _____ **Fax:** _____ **E-mail:** _____

Graphical Representation of Models

	Evaluation Criteria					
Issue	Excellent	Very Good	Good	Fair	Poor	Very Poor
Detail						
Appearance						

Detail: Degree of detail of scaffolding components (e.g. dimension and size).
Appearance: Appearance of scaffolding components compared to a real-life appearance.

System Performance

	Evaluation Criteria					
Issue	Excellent	Very Good	Good	Fair	Poor	Very Poor
User Interface						
Problem/Solution Representation						
Applicability for Training						
Overall Performance						

User Interface: User friendliness of system environment.
Problem/Solution Representation: Clarity and correctness of scaffolding problems and solutions.
Applicability for Training: Usefulness and practicality of the system.

Figure 7-1. System evaluation form.

(1) How interested are you in using SAVR for training construction workers or other safety-related personnel? Please circle/underline your answer

Very interested Interested Neutral Not so interested Not at all

(2) What benefits (if any) can SAVR bring to your company/organization (e.g. teaching new workers how to detect a hazard in a scaffolding platform)?

(3) Could you furnish your opinion regarding the potential impact of using SAVR in the construction industry.

(4) Based on your experience, could you list the critical needs for safety training for the next five years.

(5) What role should we (the academic community) play in addressing these needs?

Figure 7-1. System evaluation form (continued).

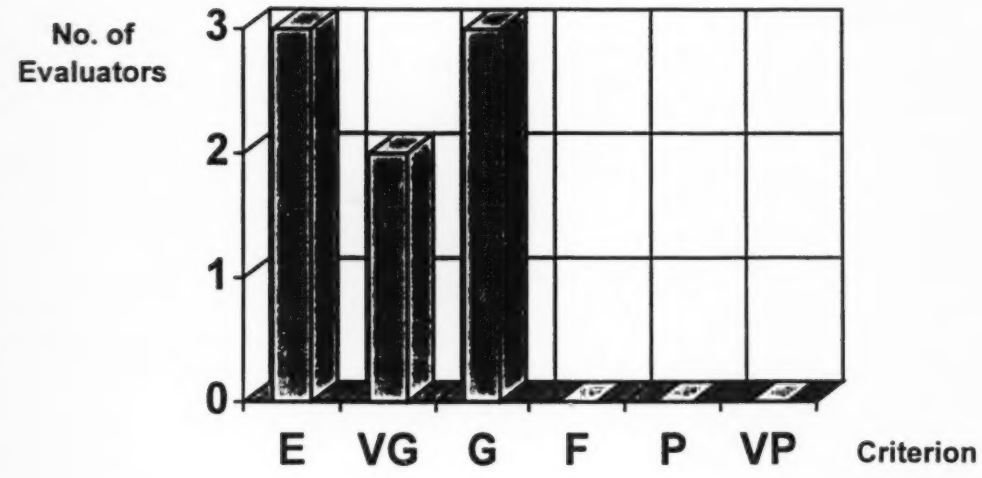

Figure7-2. Detail of graphical models.

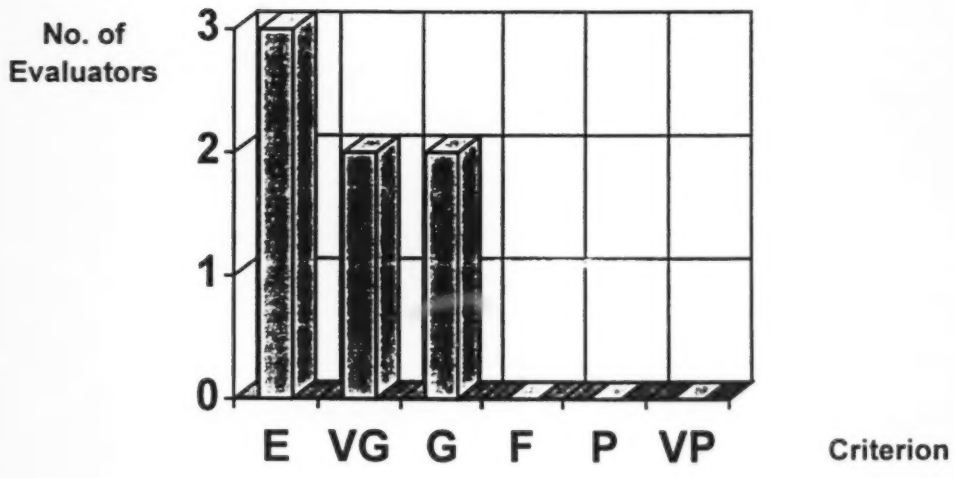

Figure 7-3. Appearance of graphical models.

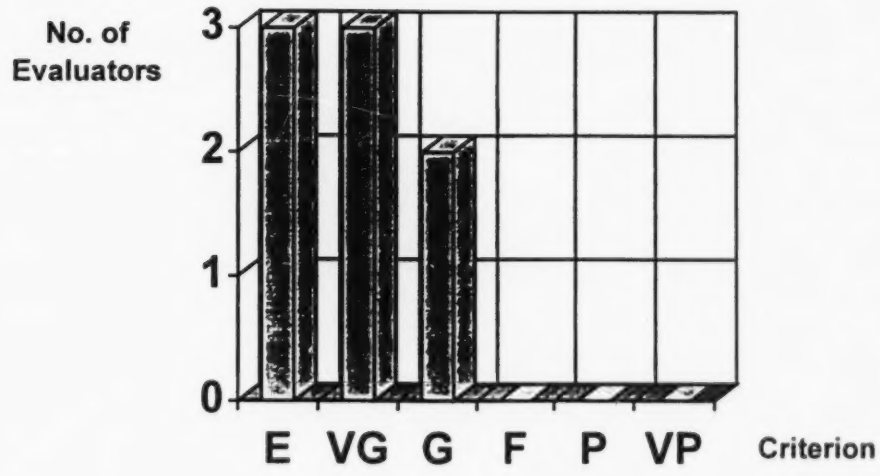

Figure 7-4. The system's User Interface.

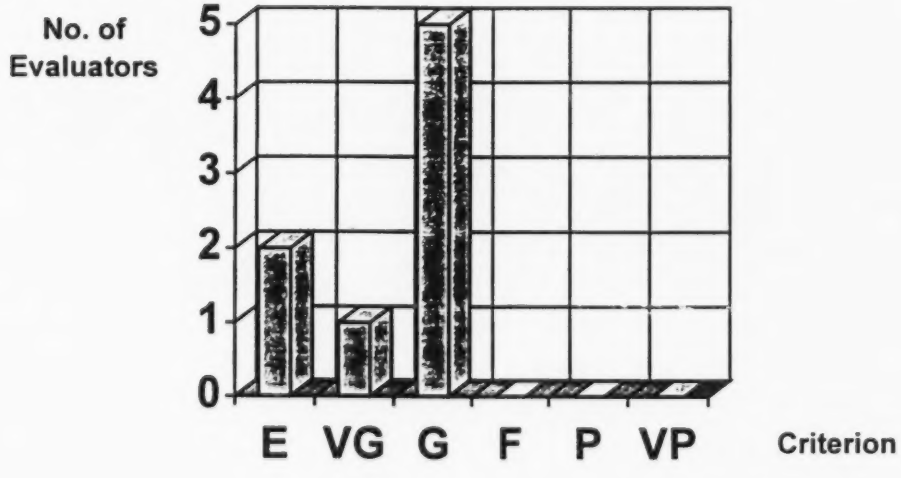

Figure 7-5. The system's problem/solution representation.

BEST COPY AVAILABLE

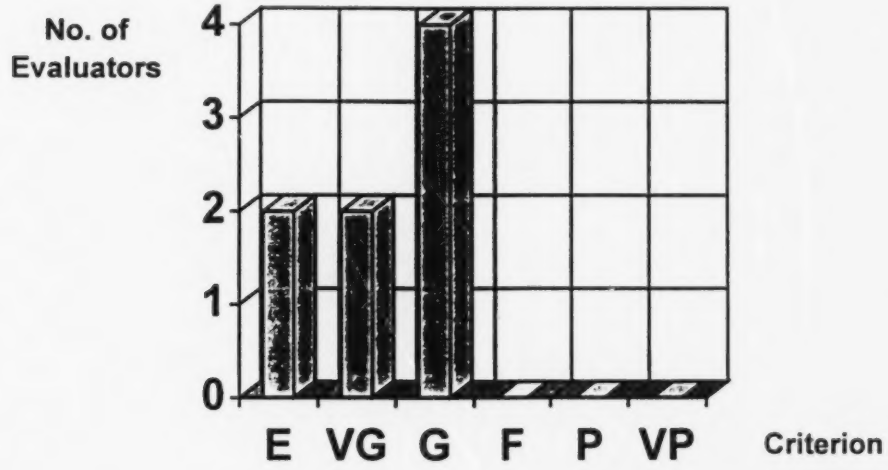

Figure 7-6. The system's applicability for training.

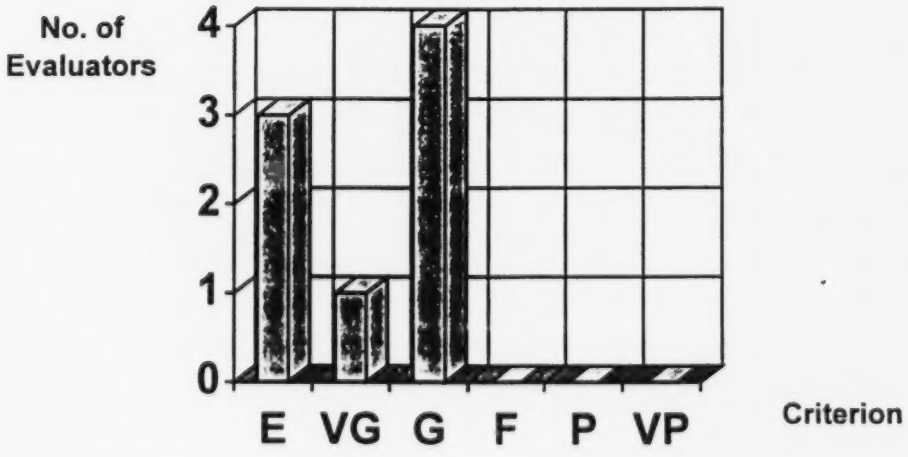

Figure 7-7. The system's Overall Performance.

BEST COPY AVAILABLE

CHAPTER VIII

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

8.1 Summary

Construction falls result in numerous injuries and fatalities which usually lead to major losses in terms of lives, money, and time. Construction falls can be grouped into several types, such as falls from floor edges, falls from floor openings, and falls from scaffolding. In order to prevent falls, the development of an effective training tool to maintain a safe working environment is essential.

In this study, an interactive training model called SAVR was developed to train construction students, novice engineers, and construction workers to prevent falls from form scaffolding. The model was implemented in a graphics super computer, the ONYX Reality Engine2. The SAVR model provides trainees with an immersive virtual environment to perform "on the job" safety training without physically being in a real construction site. They can use a head-mounted display (HMD) and a cyberglove or a mouse to interact with the computer-generated environment.

The SAVR model includes two major training environments: erection and inspection. The erection environment is used to teach trainees the correct procedure to erect a commonly used metal bracket form scaffolding. In this environment, while trainees are immersed using the HMD, they use the cyberglove to pick and move the scaffolding components to erect the platform. An interface panel is used in this environment to provide users with erection instructions through a step-by-step procedure. After the users learn the correct erection procedure, the inspection environment introduces several hazardous conditions in an existing platform and requires trainees to visually identify them. Trainees are immersed in this environment using the HMD. Trainees have to detect each hazard using a mouse and then define its type by selecting a button from an interface panel, which includes descriptions of all existing hazards. When a hazard is correctly identified, trainees earn a score which is displayed on the interface panel.

Besides the erection and inspection environments, the SAVR model provides two supplementary environments: start and help. The start environment enables users to select among the help, erection, and inspection environments. The help environment provides users with the necessary information on how to use the program. Each environment in SAVR was implemented using a unique module and supported by several sound files.

The development of SAVR encompassed four major tasks: knowledge acquisition, model development, model validation, and reports preparation. The

knowledge was acquired from literature, previous research, experts, and safety codes. Based on the knowledge acquisition, several common potential causes of falls from form scaffolding platforms were chosen for the SAVR model, focusing on the causes which could be visually presented. These causes included three major groups of scaffolding problems: component problems, connection problems, and other miscellaneous problems. Each group included several problems related to one or more scaffolding components.

The second task, model development, included the construction of the 3D graphical objects of the scaffolding components, the construction of the texture images for SAVR's interface panels, and the construction of the SAVR program. In constructing the 3D objects for the scaffolding components, two major factors were considered: an object's degree of complexity, and the representation of its faces. While maintaining a realistic appearance, the graphical objects in SAVR were constructed using a minimal number of polygons to reduce their complexity. Object faces can be presented using triangular or quadrilateral polygons. In SAVR, object faces were presented using quadrilateral polygons, which reduced the total number of polygons in each object by 50%.

In constructing the SAVR program we adapted a developmental approach that included six steps: (1) defining the problem, (2) designing the solution, (3) refining the solution, (4) considering a testing strategy, (5) coding, testing and debugging the program, and (6) documenting the program. In using this approach, our programming considerations involved the programming environment, the structure of the program, the flow of the program, the graphical user interface, and the expansion of the application. During every developmental stage, we gave high priority to developing a user-friendly graphical-user interface.

8.2 Conclusions

This study has resulted in the SAVR prototype, an interactive and immersive training tool which utilizes virtual reality technology for safety training. SAVR demonstrates the potential of virtual reality technology in safety training. To illustrate, SAVR can be used to teach trainees how to properly erect a metal bracket of a form scaffolding. In addition, SAVR can test their ability to identify many potential fall causes related to this scaffolding type. Furthermore, SAVR provides a safe environment for worker training.

We expect that the interactive visualization in the SAVR model will enable trainees to visually memorize the erection and inspection patterns and consequently, apply them in real-life situations. Even though SAVR demonstrates potential fall causes for a particular scaffolding platform, most of these potential fall causes are applicable to other types of scaffolding. For example, the component problems presented in SAVR for the guardrail system, which involve missing and defective components, are typical problems of guardrail system in other types of scaffolding platforms. When such typical problems are found in other scaffolding platforms, trainees can identify and eliminate them.

The interactive visualization in SAVR called for an extensive use of graphics due to the complexity of the 3D objects presenting the steel components of the scaffolding. While these objects were constructed using a minimal number of polygons, they still involved complex graphics which require advanced rendering capabilities. Accordingly, SAVR was developed on a graphics super computer to facilitate a real time interaction. This real-time interaction is achieved in most cases during the simulation, where the frame rates exceed 25 frames per second and can even reach 58 frames per second. However, in some cases the frame rates are less than 25 frames per second. This is due to the complexity of the graphics in some particular scenes which could not be further simplified without ignoring necessary geometrical detail.

The modular design for implementing the start, help, erect and inspect environments in the SAVR model enables future expansion of the model to include additional types of scaffolding platforms. In addition, this modular design helped in avoiding a significant decrease of the frame rates, which would occur if all environments (help, start, erection, and inspection) were included in one module. Moreover, this modular design simplified the debugging process during application development.

The implementation of an interface panel in each module and the use of textures for panel functions provided a user-friendly graphical-user interface. The use of the head-mounted display and the cyberglove for user interaction with the SAVR program enhanced the program interactiveness and immersiveness. While using the cyberglove simplifies the training process, its implementation was time consuming. This is due to the complexity of coding the interface scenarios, which required the program to detect user gestures and consequently respond to each one during the simulation.

In developing SAVR, we have been successfully translated the model from Onyx RE2 to a PC platform. The PC we used was a Pentium 60 with 16 MB RAM. While it is still acceptable, understandably, the performance of the model cannot be compared with that when it were run on the Onyx RE2. The performance of the model is expected to improve considerably if we used the model in an enhanced PC environment (e.g., with Pentium Pro 200, 32 MB RAM, and a powerful graphics accelerator). Note also that the capacity of PCs doubles every 18 months, hence, in not too distant future a much enhanced PC will narrow the VR application gap between Windows and Unix-based operating systems.

Throughout the project period (12 months), SAVR was continuously tested and evaluated by its developers (investigators and his student researcher), particularly for its erection and inspection modules. After its completion, SAVR's inspection module was formally tested by eight experts from various companies and safety organizations. The experts' assessment of model detail and appearance suggests that, in their eyes, the model seems to be sufficiently realistic. Further, the model's overall performance as a training tool was judged close to very good. Based on the results of the experts' evaluation we conclude that this study has reached its objectives.

Comments from experts further indicate that when SAVR is implemented it can become a self-contained training tool for entry level construction laborers. The usefulness

of SAVR as a training tool can be extended to include construction students. In the near future, the system is expected to contribute to the avoidance of construction falls, and subsequently, to the reduction of injuries and fatalities of construction laborers during construction operations.

8.3 Recommendations

Based on our experience in SAVR development, we recommend incorporating other types of hazardous construction platforms (roofs, floor holes/openings, floor edges, wall openings, tops of walls, ladders, steel beams, and other types of scaffoldings) for worker training. This expansion is currently feasible due to the modular structure of the SAVR program. Each type of hazardous platforms can be incorporated into the SAVR structure using a separate module. Also, other types of scaffolding, such as tube and coupler, suspended, wood pole, tubular welded, and mobile scaffoldings are potential extension modules of SAVR. The knowledge base required to incorporate these types has already been acquired through the research done by Hadipriono et al. [1995c].

APPENDIX A

REFERENCES

Adams, J. M., Philippe, J. G., and Barry L. K. (1988). "An Introduction to Computer Science with Modula 2.," D. C. Heath and Company.

AGC (1990). "AGC Guide for A Basic Company Safety Program." Published by The Associated General Contractors of America, Washington, D.C.

Ann, H.S. (1993). "Safety Consciousness of Construction Manager," A Report presented to the Korean Industrial Safety Institution, Inchon, Korea.

Aukstakalnis, S. and Blatner, D. [1992]. "Silicon Mirage: The Art and Science of Virtual Reality." edited by Roth, S.F., Peachpit Press, Inc., Berkeley, California.

Aukstakalnis, S. (1993). "Go with the Flow." *Computer-Aided Design, CADalyst,* August, p. 56.

Barsoum, A. S. (1995) "The Construction of the 3D Graphical Models for the Intelligent Traffic Evaluator for Prompt Incident diagnoses in a virtual reality environment." A Thesis Presented in Partial Fulfillment of the Requirements for the Degree on Master of Science in the Graduate School of The Ohio State University.

BLS (1986a). "Injuries to Construction Laborers." Bulletin 2252. Prepared by US Department of Labor -- Bureau of Labor Statistics, Washington D.C. 20402.

BLS (1986b). "Injuries Resulting From Falls From Elevations." Bulletin 2195. Prepared by US Department of Labor -- Bureau of Labor Statistics, Washington D.C. 20402.

BLS (1988). "Occupational Injuries and Illnesses in the United States by Industry," Bureau of Labor Statistics, Bulletin 2195, Prepared by US Department of Labor -- Bureau of Labor Statistics, Washington D.C. 20402.

Bobick, T.G., Schnitzer, P.G., and Stanevich, R.L (1990a). "Investigation of Selected Occupational Fatalities Caused by Falls from Elevations," Advances in Industrial Ergonomics and Safety II, Edited by Biman Das, Taylor and Francis, pp. 527-534.

Bobick, T.G., Bell, C.A., Stanevich, R.L., Smith, D.L., and Stout, N.A. (1990b). "Analysis of Selected Scaffold-Related Fatal Falls," Proc. Human Factors Society 34th Annual Meeting, pp. 1072-1076.

CFR-29 (1988). Code of Federal Regulations 29, Labor, Part 1926, published by the Office of the Federal Register National Archives and Records Administration.

Coull, T. B. (1991). "Texture-based VR on a Desktop Computer Using World Tool Kit." *Proceedings of VR, The 2nd Annual Conference on Virtual Reality, Artificial Reality, and Cyberspace.* Edited by Helsel, S. K., Meckler Publisher, London, pp. 37-42.

Dowding, T. J. (1991). "A Self-Contained Interactive Motorskill Trainer." In *Virtual Reality Theory, Practice, and Promise.* Edited by Helsel, S. K. and Roth, J. P. Meckler Publisher, London, pp. 44-50.

ENR (1991). "Construction Injuries Climb." *Engineering News-Record*, December 16, p. 23.

Fisher, S. S. (1991). "Virtual Environments: Personal Simulation and Telepresence." In *Virtual Reality Theory, Practice, and Promise*, Edited by Helsel, S. K. and Roth, J. P. Meckler Publisher, London, pp. 101-110.

Glen, S. (1991). "Real Fun, Virtually." *Proceedings of VR, The 2nd Annual Conference on Virtual Reality, Artificial Reality, and Cyberspace.* Ed. by Helsel, S. K., Meckler Publisher, London, pp. 62-69.

Goldman, G. and Zdepski, M. S. (1991). "Reality and Virtual Reality." Publisher: Association for Computer Aided Design in Architecture (ACADIA), School of Architecture, New Jersey Institute of Technology, University Heights, Newark, New Jersey.

Hadipriono, F. C. (1985). "Analysis of Events in Recent Structural Failures." *ASCE--Journal of Structural Engineering*, Vol. 111, No. 7, pp. 1468-1481.

Hadipriono, F.C. (1992a). "Construction Visualization and Automation." *A Construction Forecast*, Builders Exchange.

Hadipriono, F.C. (1992b). "FTES-FALL: A Fault Tree Expert System for Labor Safety During Construction," *Center for Labor Research Second Annual Symposium*, The Ohio State University, Columbus, Ohio, p.1.

Hadipriono, F.C. (1992c). "A Fault Tree Expert System Model for Construction Safety Part One: Fault Tree Models," *ASCE Journal of Performance of Constructed Facilities*, Vol. 6, No. 4, pp. 246-260.

Hadipriono, F.C. (1992d). "A Fault Tree Expert System Model for Construction Safety Part Two: Fault Tree Models," *ASCE Journal of Performance of Constructed Facilities*, Vol. 6, No. 4, pp. 261-274.

Hadipriono, F.C. and Diaz C.F. (1988). "Trends in Recent Construction and Structural Failures," *Journal of Forensic Engineering*, Vol. 1., No. 4, pp. 227-232.

Hadipriono, F. C. and Larew, R. E. (1985). "Simulation Laboratory Designed to Prevent Construction-related Failures," *Engineering Education*, Vol. 76, No. 3, pp. 168-170.

Hadipriono, F. C. and Larew, R. E. (1991). "Construction Research and Practice," *Annual Conference for Engineers and Architects*, The Ohio State University, April, 1991.

Hadipriono, F.C. and Wang, H.K. (1986). "Analysis of Causes of Falsework Failures in Concrete Structures," *ASCE--Journal of Construction Engineering and Management*, Vol. 112, No. 1, pp. 112-121.

Hadipriono, F.C., Barsoum, A.S., Tsay, T.C. (1995a) "3-D Schematic Visualization for Intelligent Traffic Evaluator for Prompt Incident Diagnosis in A Virtual Reality Environment," *Invited Presentation in the fourth Conference on the Application of High Performance Computers in Engineering*, Milan, Italy, June 1995.

Hadipriono, F.C., Barsoum, A.S., Tsay, T.C., Nemeth, Z.A., and Larew, R.E. (1995b) "Visualization and Graphics in INTREPID-VR," *Proceedings of the second International Conference on Visualization and Intelligent Design in Engineering and Architecture (VIDEA 95)*, La Coruna, Spain, pp. 107-113.

Hadipriono, F. C., Vargas, C. F., and Yoo, W. (1995b). "Safety First: A Fault Tree Expert System for Construction Falls, Vol. 1: Knowledge Acquisition," *A report submitted to the National Institute for Safety and Health (NIOSH)*, Centers for Disease Control and Prevention, Atlanta, GA.

Heilig, M. (1962). "Sensorama Simulator." US Patent No. 3,050,870, August 28.

IC (1994). "Safety Record," International Construction Vol. 33, No. 5, p. 24.

Jacobson, L. (1994). "Garage Virtual Reality, The Affordable Way to Explore Virtual World," First Edition, Sams Publishing, A Division of Prentice Hall Publishing, Indianapolis, IN 46290.

McClarnon, D. (1995). "tri-quad.txt." A text file downloaded from the WTK Users' Group (SIG-WTK).

McCluskey, J. (1991). "Educational Applications of Virtual Reality: Medium or Myth?" *Proceedings of VR, The 2nd Annual Conference on VR, Artificial Reality, and Cyberspace.* Edited by S. K. Helsel, San Francisco, Meckler Publisher, London, pp. 148-153.

Nikkan Kensetsu Kougyou Shinbun (1992). "Appeals From Construction Sites in Charge of 70 Trillion Yen," a newspaper article in Japanese, March 20.

NIOSH (1988). "Fatal accident circumstances and epidemiology (FACE): Sheet metal helper falls to his death through a skylight opening in South Carolina." FACE Report 88-18, Published by US Department of Health and Human Services, Public Health Service, Centers for Disease Control, National Institute for Occupational Safety and Health, Division of Safety Research, Morgantown, WV.

NIOSH (1989a). "National Traumatic Occupational Fatality (NTOF) Data Base," Cincinnati, Ohio: U.S. Department of Health and Human Services, Public Health Service, Centers for Disease Control, National Institute for Occupational Safety and Health, Division of Safety Research, Morgantown, WV.

NIOSH (1989b). "NIOSH Alert: Request for Assistance in Preventing Worker Deaths and Injuries from Falls Trough Skylights and Roof Openings." Published by US Department of Health and Human Services, Public Health Service, Centers for Disease

Control, National Institute for Occupational Safety and Health, Division of Safety Research, Morgantown, WV.

NSC (1987). "Accident Facts." Published by National Safety Council, Chicago, Illinois.

OBWC (1979). "Specific Safety Requirements of the Ohio Bureau of Workers' Compensation relating to Construction," Ohio Bureau of Workers' Compensation, Division of Safety and Hygiene, Columbus, Ohio.

OBWC (1989a). "Specific Safety Requirements of the Ohio Bureau of Worker's Compensation relating to Construction." Published by the Ohio Bureau of Worker's Compensation, Division of Safety and Hygiene, Columbus, Ohio.

OBWC (1989b). "Ohio Falls and Slips (Type 4, 5, 10)," Construction Industry and Occupation (Public and Private Sector) 1989 Injury/Illness Statistics. Prepared by The Occupational Health and Safety Research Section, Ohio Bureau of Worker's Compensation, Division of Safety and Hygiene, Columbus, Ohio.

Pimentel, K. and Teixeira, K. (1993). "Virtual Reality, Through the New Looking Glass," First Edition, Windcrest Book Publishing, A Division of McGraw-Hill Inc., New York, NY.

Rheingold, H. (1991). *Virtual Reality.* Summit Books Publisher, New York, New York 10020.

SGI. (1994). "Onyx and Power Onyx Product Guide." Silicon Graphics Inc., Mountain View, California.

Shaw, R. (1981). "Educational Requirements for Avoidance of Structural Adequacy," In *Structural Failures in Buildings*, The Institution of Structural Engineers, London.

TBR. (1989a). "Model for A Constructor Safety Process," Construction Industry Safety Excellence Award Program, *The Business Roundtable*, New York, NY, 10166.

TBR. (1989b). "Model for An Owner Safety Process," Construction Industry Safety Excellence Award Program," *The Business Roundtable*, New York, NY, 10166.

TBR. (1990). "Improving Construction Safety Performance," A Construction Industry Cost Effectiveness Project Report, *The Business Roundtable*, Report A-3 1990, New York, NY, 10166.

Vargas, C.A. (1993). "Construction Falls: Knowledge Acquisition and Fault Tree Development," a thesis presented in partial fulfillment of the requirements for the degree of Master of Science in the graduate school of the Ohio State University.

Vargas, C.A. and Hadipriono, F.C. (1995). "Knowledge Acquisition and Knowledge Structure for SAFETY FIRST System," *Proceedings for the ASC National Conference, Phoenix, Arizona,* May 1995, pp. 227-235.

Walker, A. C. (1981). "Study and Analysis of the First 120 Failure Cases." In *Structural Failures in Buildings*, The Institution of Structural Engineers, London.

WTK. (1995). "World Tool Kit for SGI Version 2.1 Manual." Sense 8, Inc., California.

Yoo, W.H. (1994). "Fault Tree Modeling of Construction Falls from Scaffolding," a thesis presented in partial fulfillment of the requirements for the degree of Master of Science in the graduate school of the Ohio State University.

APPENDIX B
LIST OF ABBREVIATIONS

2D	**Two-Dimensional**
3D	**Three-Dimensional**
3DS	The default file-name extension for **3D** Studio files
ASCE	American Society of Civil Engineering
BMP	The file-name extension for **Bitmap** files
CLASS	Construction Laboratory for Automation and System Simulation
CLR-OSU	Center for Labor Research at the Ohio State University
COVR	Construction Operations Using Virtual Reality
EFCO	Economy Forms Co.
HMD	Head Mounted Display
INTREPID-VR	INtelligent TRaffic Evaluator for Prompt Incident Diagnosis in a Virtual Reality environment
NFF	Neutral File Format
NIOSH	National Institute for Safety and Health
NTOF	National Traumatic Occupational Fatality
NSF	National Science Foundation
OBWC	Ohio Bureau of Workers' Compensation
Onyx RE2	Onyx Reality Engine 2
OSHA	Occupational Safety and Health Administration
OSU-CITR	The Ohio State University Center of Intelligent Transportation Research
PC	Personal Computer
R&D	Research and Development
RGB	The SGI default file-name extension for image files
SAVR	Safety in Construction Using Virtual Reality

SGI	Silicon Graphics Inc.
TGA	The file-name extension for **Targa** image files
TXT	The file-name extension for **Text** files
VB	Visual Basic
VR	Virtual Reality
WS-FTP	Windows' Socket File Transfer Protocol
WTK	World Tool Kit

CENTER FOR LABOR RESEARCH
THE OHIO STATE UNIVERSITY

WORKING PAPER SERIES

WP-001 **The Workers' Contribution to the Workers' Compensation Quid Pro Quo: Broad or Narrow?** — Professor Deborah A. Ballam, Faculty of Finance (December, 1991)

WP-002 **Economic Interest Groups and Elections to the Ohio Supreme Court, 1986 and 1988** — Professor Lawrence Baum and Ms. Marie Hojnacki, Department of Political Science (December, 1991)

WP-003 **Employment-Based Training in Japanese Firms in Japan and in the United States: Experiences of Automobile Manufacturers** — Professor Masanori Hashimoto, Department of Economics (February, 1992)

WP-004 **Organized Labor and Political Action, Attitudes, and Behavior** — Professor Herbert B. Asher, Professor Randall B. Ripley, and Ms. Karen C. Snyder, Department of Political Science (October, 1992)

WP-005 **Organizational Pay Systems: The Union's Role in Promoting Justice, Satisfaction, and Commitment to the Union** — Professor Marcia P. Miceli, Professor Susan L. Josephs, and Mr. Matthew C. Lane, College of Business; and Mr. Paul W. Mulvey, University of Connecticut (October, 1992)

WP-006 **Challenging the Roadblocks to Equality: Race Relations and Civil Rights in the CIO 1935—1955** — Professor Marshall F. Stevenson, Jr., Department of History (December, 1992)

WP-007 **The Economic Impact of Development: Honda in Ohio** — Professor Mary K. Marvel and Professor William J. Shkurti, School of Public Policy and Management (December, 1992)

WP-008 **Union-Management Cooperation: A Process for Increasing Worker Autonomy and Improving Work Group Effectiveness?** — Professor Philip R. Kroll, Agricultural Technical Institute; Professor Stephen J. Havlovic and Professor Gervase Bushe, Simon Fraser University (December, 1992)

WP-009 **A Cross-Disciplinary Integrative Summary of Research on Workplace Substance Abuse** — Professor David A. Smith, Department of Psychology (December, 1992)

WP-010 **A History of Labor in Columbus, Ohio 1812—1992** — Professor Warren R. Van Tine, Department of History (December, 1993)

WP-011 **Labor and the Mass Media: A Case Study and Survey of Secondary Literature** — Elizabeth A. Daley, Department of Communication (June, 1994)

WP-012 **Benefits, Unions and Work-Family Time Conflict** — Professor Toby L. Parcel, Department of Sociology (October, 1994)

WP-013 **Workplace Innovation and Local Unions in the Building Trades: Theory, Application and Membership Reactions** — Professor Marcus Hart Sandver and Mr. Jeffrey A. Miles, Department of Management and Human Resources (October, 1994)

WP-014 **Public Sector Collective Bargaining In Ohio, 1984—1993: A Statitstical Overview** — Mr. Daniel E. Ashyk, Center for Labor Research (February, 1995)

WP-015 **Local Economic Development, Tax Abatement, and the Role of Self-Government in Large U.S. Cities** — Professor Charles F. Adams and Mr. Im-Gon Cho, School of Public Policy and Management (February, 1995)

WP-016 **Youth, Taxes and Pension Coverage** — Professor Patricia A. Reagan, Department of Economics, and Mr. John A.Turner, U.S. Department of Labor (April, 1995)

WP-017 **Davis-Bacon Compliance and Enforcement Programs** — Professor Marcus Hart Sandver, Department of Management and Human Resources (April, 1995)

WP-018 **The Metropolitan Contingency of the Male Youth Central-City Employment Disadvantage** — Professor Steven R. Holloway, Department of Geography (August, 1995)

WP-019 **The Influence of Organized Labor on U.S. Policy Toward Israel, 1945-1967** — Professor Peter L. Hahn, Department of History (February, 1996)

WP-020 **Labor Market Performance of Non-College-Bound Youths** — Professors Masanori Hashimoto and Ross A. Miller, Department of Economics (February, 1996)

WP-021 **Judicial Attitudes Toward Sexual Harassment in the Workplace** — Professor L. Camille Hebert, College of Law (February, 1996)

WP-022 **SAfety in Construction Using Virtual Reality (SAVR): A Model for Labor Safety** — Professor Fabian C. Hadipriono, Professor Richard E. Larew, and Ashraf S. Barsoum, Department of Civil and Environmental Engineering and Geodetic Science (June, 1996)

University Grants Committee

Warren R. Van Tine, Chair
John T. Demel
Johanna S. DeStefano
Richard J. First
Stephen F. Loebs
Stephen L. Mangum
Toby L. Parcel
Edward J. Ray
Randall B. Ripley
Nancy A. Rogers
Ray D. Ryan, Jr.

Statutory Advisory Committee

William A. Burga, Chair
Donald K. Day
Dan Winslow
Jeffrey A. Rechenbach
Paul J. Witte

Administration

Edward F. Hayes
Vice President for Research
C.J. Slanicka,
Director
Sandra L. Jordan,
Assistant Director

CLR
Center for Labor Research
1314 Kinnear Road, Room 204
Columbus, Ohio 43212-1194
Phone 614 292-4440

BEST COPY AVAILABLE